AFRL-IF-RS-TR-2005-281
Final Technical Report
August 2005

NEXT GENERATION (XG) ARCHITECTURE AND PROTOCOL DEVELOPMENT (XAP)

BBN Technologies

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. P858

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-281 has been reviewed and is approved for publication

APPROVED: /s/

RICHARD D. HINMAN
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	AUGUST 2005	Final Jun 03 – Dec 04

4. TITLE AND SUBTITLE
NEXT GENERATION (XG) ARCHITECTURE AND PROTOCOL DEVELOPMENT (XAP)

5. FUNDING NUMBERS
C - F30602-03-C-0139
PE - 63760E
PR - P858
TA - XG
WU - II

6. AUTHOR(S)
Ram Ramanathan and Craig Partridge

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
BBN Technologies
10 Moulton Street
Cambridge Massachusetts 02138

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFGC
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-281

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Righard D. Hinman/IFGC/(315) 330-3616/ Richard.Hinman@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
The DARPA XG program seeks to develop and demonstrate adaptive spectrum utilization techniques which can be used to access unused spectrum without interference to incumbent users. This report describes the progress which was made to develop an XG architecture which is capable of accommodating a wide range of adaptive protocols operating within the bounds of spectral policy. Additionally this development included the definition of a machine readable policy language and a policy conformance reasoner to validate if candidate transmissions are compliant to spectrum policy.

14. SUBJECT TERMS
Adaptive Communication, Cognitive Radio, XG, Adaptive Spectrum Utilization, Machine Readable Policy

15. NUMBER OF PAGES
248

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

List of Tables

1 Scope

This is a report on the *XG Architecture and Protocol Development (XAP)* project. The XAP project was funded under the DARPA neXt Generation (XG) Communications program (Phase 2), beginning in June 2003, and ending in December 2004. XAP is a follow on to the *Medium Access Control Protocols for XG* (X-MAC) project that was funded under the DARPA XG program (Phase 1).

This report contains an overview of the project accomplishments. We emphasize that this report is only an overview, and details can be found in a number of supporting documents and Request for Comments (RFCs) contained in the XAP distributions, including:

1. The *XG Vision RFC* [XGV], which describes the vision and goals of the XG program in general and X-MAC in particular.
2. The *XG Architecture RFC* [XGAF], which presents the architecture, system components, and a high level concept of operations for XG communications.
3. The *XG Abstract Behaviors RFC* [XGAB], which identifies key behaviors that must be implemented by an XG system, organizes them, and describes the behaviors.
4. The *XG Policy Language Framework RFC* [XGPLF], which describes the policy specification meta-language for implementing machine-understandable policies.
5. The *XG Policy Conformance Reasoner* software and documentation [PCR], which describes a reference implementation based on the XG Policy Language Framework RFC v1.0.
6. The *XG Evaluation Platform* [XEP], which describes the OPNET models of XG protocols and simulation results.

Accomplishments under XAP can be classified into two broad categories – (a) the development of a framework for managing the key aspects of radio behavior through flexible application of policies, (b) design, modeling and simulation of key protocols for opportunistic spectrum access. Items (1)-(5) above fall under the framework, and (6) under the protocol modeling and simulation. For most of the project duration, these two sets of activities were conducted in parallel.

The remainder of this document is organized as follows. In Section 2, we present the background and motivation for XAP. Further, we describe the XG vision and architectural framework that was developed largely in the earlier XMAC contract, and has been refined during the XAP effort through the working group process. In the XAP effort, we have developed specific technologies that can enable policy-agile opportunistic spectrum access by XG systems. These include the policy language framework and policy conformance reasoner software that we describe in Section 3, the abstract behavior specification that we describe in Section 4, and the XG evaluation platform and protocol simulation results that we describe in Section 5. In Section 6 we discuss technical interactions related to the XG program both within the XG program including working group activities as well as external interactions including briefings to government, academia and industry. In Section 7, we conclude with a summary of lessons learned from the project, a roadmap for the XG policy-agile opportunistic spectrum access technology (status at the end of Phase 2 and projections for Phase 3 and beyond), and recommendations for future work.

2 Background

There are two significant problems confronting wireless communications with respect to spectrum use:

- *Scarcity*. The current method of allotting spectrum provides each new service with its own fixed block of spectrum. Since the amount of useable spectrum is finite, as more services are added, there will come a point at which spectrum is no longer available for allotment. We are nearing such a time, especially due to a recent dramatic increase in spectrum-based services and devices.
- *Deployment difficulty*. Currently, extensive, frequency-by-frequency, system-by-system coordination is required for each country in which these systems will be operated. As the number, size, and complexity of operations increase, the time for deployment is becoming unacceptably long.

Both problems are a consequence of the centralized, static nature of current spectrum allotment policy. This approach lacks the flexibility to aggressively exploit the possibilities for dynamic reuse of allocated spectrum over space and time, resulting in very poor utilization and *apparent* scarcity. It also mandates a priori assignment of spectrum to services before deployment, making deployment difficult.

Preliminary data indicates that large portions of allotted spectrum are unused (refer the Spectrum Policy Task Force report [SPTF]). This is true both spatially and temporally. That is, there are a number of instances of assigned spectrum that is used only in certain geographical areas, and a number of instances of assigned spectrum that is only used for brief periods of time. This wastage of assigned spectrum is bound to increase in future – spatially, due to the increasing localization of propagation due to radio devices using higher frequencies, and temporally due to the proliferation of services that are highly bursty in nature.

Studies have determined that even a very straightforward reuse of such "wasted" spectrum can provide an order of magnitude improvement in available capacity. It can be concluded that the issue is not so much that spectrum is scarce, but that we do not have the technology to effectively manage access to it in a manner that would satisfy the concerns of current licensed spectrum users.

XG (neXt Generation Communications) is a DARPA-funded research program based on the (now generally accepted) premise described above that the historic (and current) method of authorizing spectrum use–static, administrative allocation–results in an apparent scarcity of spectrum that can be avoided by the proper application of dynamic spectrum sharing techniques. The goals of the XG program are:

1. Demonstrate through technological innovation the ability to utilize available (unused, as opposed to unallocated) spectrum more efficiently.
2. Develop the underlying architecture and framework required to enable the practical application of such technological advances.

The XG program had several performers, including BBN. The goal of the other performers was to develop opportunistic spectrum access technologies. BBN's role within the program was unique – we served as the developer of a framework within which such technologies could evolve.

Specifically, XAP had two main technical goals (and accomplishments):

1. *Develop a long-lived framework for managing the key aspects of radio behavior through flexible application of policies*. In order that the radio is policy-agile, we require a framework in which policies are written in a way that can be interpreted by the radio, and the radio is able to exploit such expression of policies. Furthermore we require a specification of key behaviors of an XG system so as to provide a common set of abstractions for policy administrators and XG system designers.

2. *Develop protocols for XG communications and analyze them using modeling and simulation*. The development of protocols gives insight into the right framework, and modeling captures the performance variations as a function of various system and environmental parameters. Further, our simulation system serves as a "proxy XG radio" in the framework.

BBN Technologies also coordinated an XG Working Group comprised of Government and support staff, and the XG Phase 2 program participants.

As part of the second goal the XG program, participants (known collectively as the XG Working Group or XG-WG) produced a series of Requests for Comments (RFC) that together describe the proposed XG architecture and framework. These documents are RFCs because the authors recognize that the final development of such an important technology cannot be accomplished by a small group of individuals. It requires input and participation from a broad representation of the affected community. Thus it is hoped these RFCs will spur that community to provide feedback in order to assure an organized and technically valid approach to the evolution of this architecture. BBN was the primary developer of all the RFCs to date, with inputs from DARPA, AFRL, and the other XG performers.

The XG Vision RFC provides the motivation and scope of dynamic spectrum sharing envisioned by the XG program and describes an approach for developing the XG architecture. It is highly recommended that the XG Vision RFC be read before reading the others. The XG Architectural Framework RFC presents the XG architecture, system components, and a high level concept of operations for XG communications. The XG Policy Language Framework was the third to be released. The draft XG Abstract Behaviors RFC is being released (to the XG program participants) at the same time as this report. It provides an abstract view of the operation of XG radios, so that policies may be written to govern the operation of radios that are yet to be engineered.

In order to address the scarcity and deployment difficulty problems, XG is pursuing an approach wherein static allotment of spectrum is complemented by the opportunistic use of unused spectrum on an instant-by-instant basis, in a manner that limits interference to primary[1] users. In other words, the basic concept of operation is as follows: a device first "senses" the spectrum it wishes to use and characterizes the presence, if any, of primary users. Based on that information, and regulatory policies applicable to that spectrum, the device identifies spectrum opportunities (in frequency, time, or even code), and transmits in a manner that limits (according to policy) the level of interference perceived by primary users. We term this approach *opportunistic spectrum access.*

Opportunistic spectrum access also provides far easier deployment, or rapid entry, into regions where spectrum has not been assigned. Only minimal prior coordination is necessary, greatly easing the restrictions to meet the deconfliction requirements. This is helpful both in civilian applications such as the entry of a wireless LAN technology in less developed regions, and in military operations requiring high tempo and quick reaction time.

2.1 The XG Vision of Policy Agility

The realization of opportunistic spectrum access is highly challenging. Several problems must be solved: sensing over a wide frequency band; identifying the presence of primaries and characterizing available opportunities; communication among devices to coordinate use of identified opportunities; and most importantly, definition and application of interference-limiting policies, and utilization of the opportunities while adhering to such policies.

The true potential of this new approach can be exploited only if in addition to spectrum agility, we provide *policy agility* – that is, a way by which the policies controlling the behavior can be dynamically

[1] Users that are licensed to use the spectrum in question, subject to regulatory constraints are called primary users.

changed. That is, policies are not embedded in the radio, but can be loaded "on-the-fly". Policy agility allows adaptation to policies changing over time and geography. Further, technology (spectrum agility) can be developed in advance of policies. This is important for breaking the chicken-and-egg dilemma that exists today, where regulatory bodies must wait for technology before drafting policies and technology must wait to see what the polices will look like.

The use of policy agility using *machine understandable* policies is depicted in Figure 1. Starting from the left, spectrum policies are encoded in a machine interpretable form and loaded into the XG device. The XG device then operates in accordance with its interpretation of these policies. Policies may be loaded using smart media or over the Internet. In order to change the policies we simply need to load a new version. For instance, operating in a different country would require merely downloading from a different website or new smart card.

Although recent years have seen some of the components for opportunistic spectrum access mature (e.g. software radios), we are a long way from a prototypical system. Further, no work exists in the area of decoupling the policies from the implementation. This yawning gap between current state-of-the-art and what is required for opportunistic spectrum access is the motivation behind XG.

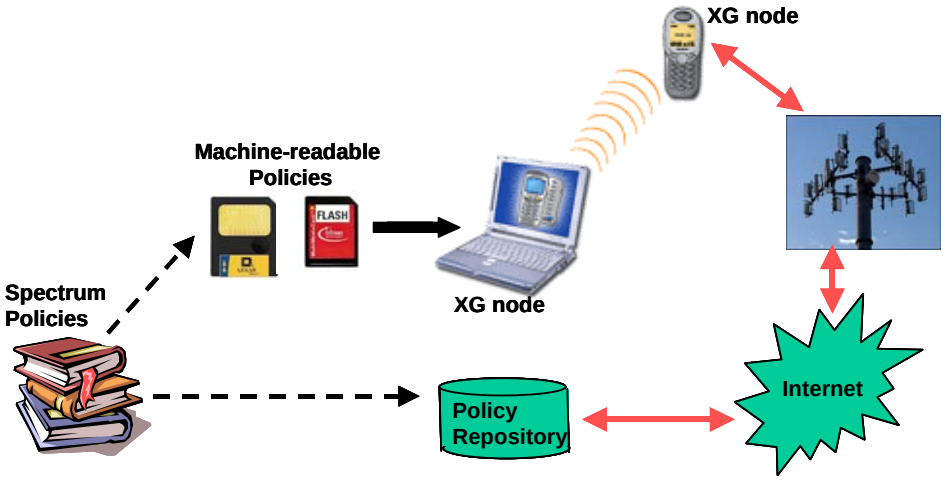


Figure 1: Machine understandable policies. With this, changing the policy merely requires loading a different flash card or downloading anew.

A more comprehensive discussion of the XG vision and motivation can be found in the XG Vision RFC [XGV].

2.2 The XG Architectural Framework

The XG framework is based on a decoupling of three fundamental elements in the design of an XG system: *policies, behaviors, and protocols.* By this, we mean that policies, behaviors and protocols are developed separately with some kind of "connections" defined between them. This concept is illustrated in Figure 2.

Decoupling allows adaptation to policies that vary over time and geography. Technology can be developed in advance of policies, and worldwide deployment would be greatly simplified. Furthermore, sub-policy management, as required for secondary spectrum markets, is easier. Policies no longer have to reflect the common denominator of competing technologies, and can be tailored to the diverse system capabilities expected for opportunistic spectrum access. Decoupling behaviors from protocols allows us to control *what* needs to be done separately from *how* it is implemented, resulting in a cleaner and more flexible architecture. Indeed, we argue that a decoupling approach is not just beneficial but pretty much a requirement for harnessing the full potential of opportunistic spectrum access.

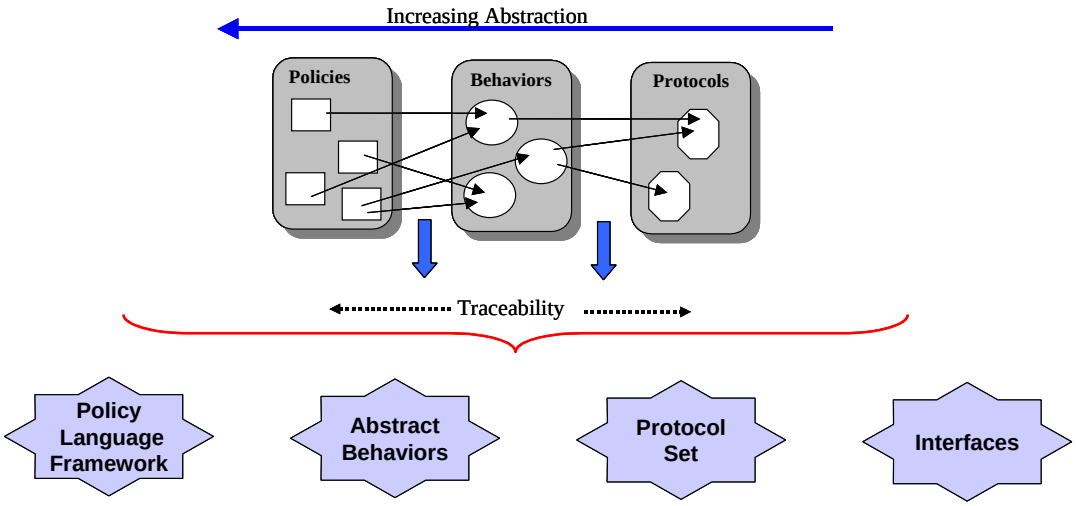


Figure 2: This illustrates the decoupling of policies, behaviors, and protocols, along with traceability and the four components of our framework.

A central notion in this approach, and a notion that is enabled by this approach, is that of *traceability*. Behaviors, preferably one or more of a core set of abstract behaviors, should be traceable to policies. This provides two advantages: first, it helps make the verification of new policies easier, and second, when an XG radio is deployed in a new region, it is easier to affirm that the XG system will behave in a certain way. It allows us to accredit based on ability to correctly interpret and implement policies.

As shown in the figure, the task of decoupling policies, behaviors and protocols requires the definition of four components in the framework: policy language, abstract behaviors, protocols, and interfaces. We describe below our accomplishments on the definition of the policy language, abstract behaviors, and one interface – namely the policy language interface. Protocols were only developed as part of the parallel modeling and simulation effort within BBN (that is, unlike the language, behaviors and policy interface, this did not have program-wide scope and consequently did not involve participation of the XG working group members, namely, the Government and performers). We give a very brief overview of some of these protocols in Section 5.

We reiterate that this report is only a summary. The reader should refer to the particular RFC (referenced in the respective section) for details. We also note that the development of the framework is an evolving process and this report reflects the current snapshot in that evolution. The follow on projects in Phase 3 will further evolve the framework, and indeed, may modify some of the definitions given here.

The XG Vision and Architectural Framework summarized in this section were largely developed within the earlier BBN XMAC effort in Phase 1 of the XG program. These were specified in the form of two RFCs [XGV, XGAF], which we have refined during the XAP effort through the working group process.

The XAP effort builds upon and significantly extends the work from the XMAC effort. We have developed specific technologies to enable policy-agile opportunistic spectrum access by XG systems. These include the policy language framework (policy ontologies and the policy conformance reasoner software) [XGPLF], the abstract behavior specification [XGAB], and protocol simulation results [XEP].

3 Policy Language Framework

XG-enabled radios will be able to utilize available spectrum intelligently based on knowledge of actual conditions rather than using current conservative spectrum management methods (static spectrum assignments). In this way, XG technologies will utilize spectrum in a much more efficient manner than can be done today. Furthermore, without the need to statically allocate spectrum for each use, networks can be deployed much more rapidly. Today the military, for example, must make spectrum use requests to their spectrum managers, who must deconflict them, and make static assignments far in advance of deployment. With XG capabilities, this process can be significantly shortened or perhaps even eliminated beyond decisions made at the highest level.

Radios must adhere to rules that apply to their operation. A major intent of such rules is to avoid or reduce interference among users. Such rules may cover both transmission and reception functions of a radio. Currently such rules are enumerated in spectrum policy as produced by various spectrum authorities such as the FCC or the NTIA in the U.S.A. In this document we use the term *spectrum policy* to refer to any externally (to the radio) imposed rules for spectrum use.

A radio that is capable of dynamically utilizing spectrum must be able to adhere to rules corresponding to the many uses of which it is capable–not just one use as with most current radios. XG radios will be expected to operate over a wide range of frequencies and within different geopolitical regions. Therefore, they must incorporate a real-time adaptive mechanism for conforming to the policies applicable to each situation. In other words, XG radios must be *policy-agile*, by which we mean both that the radios are situationally adaptive to the current policy, and that they allow policy to be dynamically updated as well.

Spectrum policies relevant to a given radio may vary in several ways:

1. Policies may be altered over time.
2. The radio (along with its user) may move from one policy administrative domain to another (e.g. a military user may be deployed to a different country).
3. Policies may be dependent on time of day or year.
4. A spectrum owner/leaser may impose policies that are more stringent than those imposed by a regulatory authority.
5. The spectrum access privileges of the radio may change in response to a change in radio user.

To be truly versatile, the XG radios should be responsive to such changes in policy.

Current radios support only a small number of modes of operation and a limited range of intended operating environments. With only limited hardware agility, all the relevant policy sets that apply can be hard-coded into the radio. However, with the increasing agility and programmability of radio hardware (as illustrated in Figure 3), especially when combined with the prospect of opportunistic spectrum sharing, both the number of modes of operation as well as the range of operating environments for the radio will increase tremendously. As a result, the number of different policy sets that apply to these various modes and environments will grow in a combinatorial fashion. This will make it impractical to hard-code into the radio discrete policy sets to cover every case of interest. The accreditation of each discrete policy set would also be a major challenge. In the case of software-defined radios, it would require the maintenance of downloadable copies of software implementations of each policy set for every radio platform of interest.

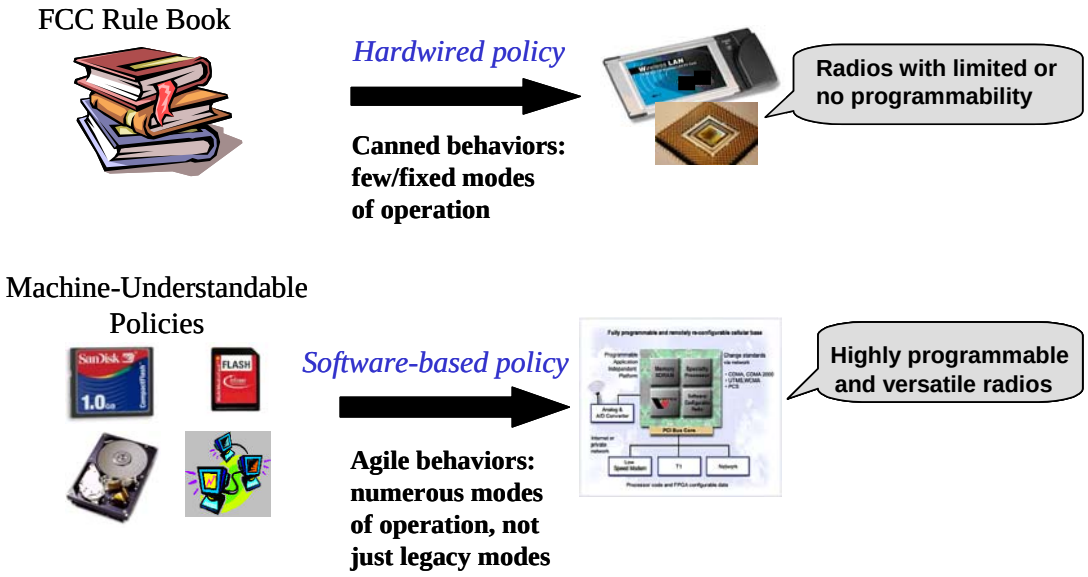


Figure 3: Machine understandable policies are necessary to exploit the emerging agility of devices and enable in-situ policy-based control of radio behaviors

We need a more scalable way to express and enforce policy. The complexity of accrediting policy conformance for XG radios and the desire for policy agility lead to the conclusion that XG radios must be able to read and interpret policy. We must, therefore, be able to express policy in a well-defined language framework.

At this point we note the distinction between the *policy language* and the *policy language framework*. We use the term *XG policy language* to refer to the language elements (including the syntactic and semantic structure of the language, and the domain-specific keywords), as well as the external (lexical) representation of the language. Overall, in this document we emphasize the language elements; we defer the lexical representation to the appendices. For this purpose, we make use of a shorthand notation to illustrate the concepts. We use the term *policy language framework* to include the policy language itself, the concept of operations of creation and use of machine understandable policies, and the computational logic, tools, and techniques for policy processing.

It is important not to confuse the policy language with its lexical representation or its syntax. We will use OWL representation (based on RDF and XML) as a basis of expression—these are not to be confused with the XG policy language. Rather, the XG policy language consists of terms (along with their precise meanings), interrelationships between the terms, some constructs and idioms that are required to express policy, and the mechanisms by which the language can be extended.

In Section 3.1we describe a concept of operations–both how machine understandable policies are created and encoded by policy administrators, as well as how the encoded policy is used by policy-agile radios. In Section 3.2, we provide a brief overview of the features of the policy language and the policy processing logic. In Section 3.3, we examine the requirements for policy language representations and we propose an extensible, standards-based machine-understandable representation for policies.

3.1 Concept of Operations

The concept of operations of the XG policy language framework describes how machine understandable policies are created and how they are used. First we will describe the actors involved,

their roles in creating and using policy, and the elements of the policy that they use or control. Then we describe how radios can interpret and use the encoded policy.

3.1.1 Development of Machine Understandable Policies

Three different types of actors are involved in the definition and use of the policy language and the policy rules: language designers, policy administrators, and spectrum users. These actors will interact with the policy language and policy rules using tools appropriate for their role as illustrated in Figure 4 and described below.

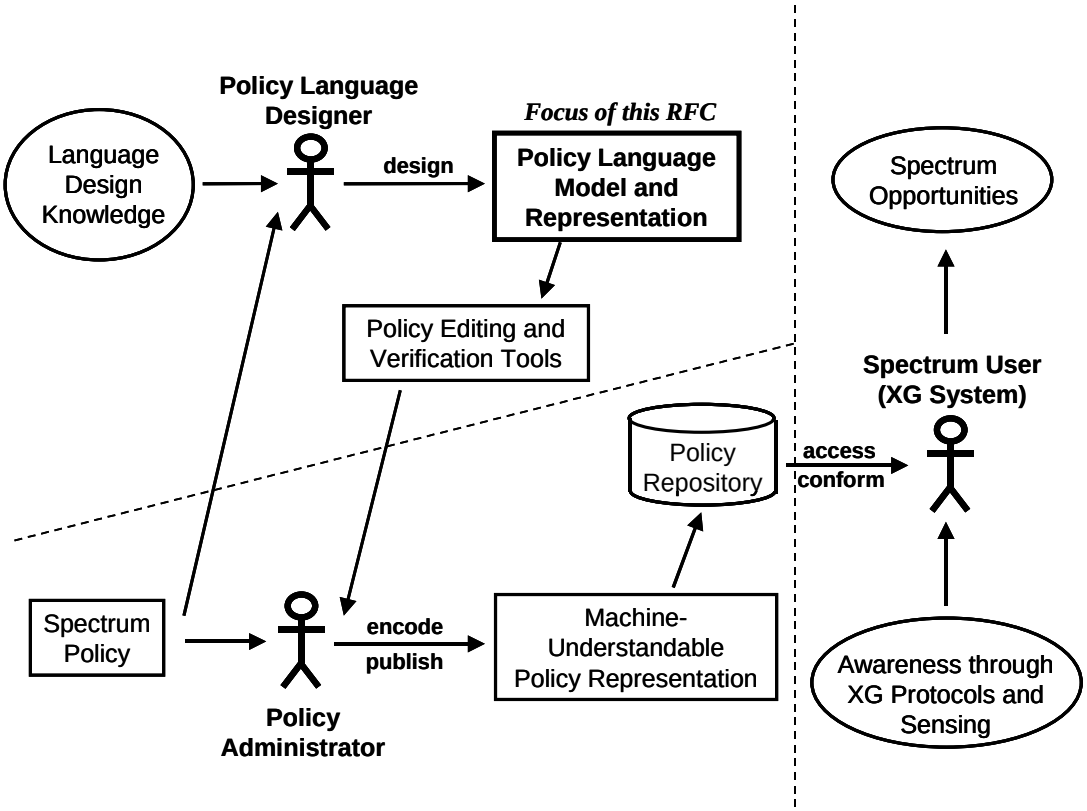


Figure 4: Policy Language Actors

Language designers create the *language model*, which defines the high-level objects of the language along with their semantics. This document describes an initial version of the language model. We expect that eventually a standards committee will draft future versions of the model. The language model is published at a well-known URI for policy administrators and policy users to access. The language designer may utilize tools (such as UML modeling tools, tools to convert UML to a machine-understandable language and other visualization tools yet to be developed) in order to facilitate the design process.

A *policy administrator* (who is not necessarily the policy maker) is responsible for developing and encoding spectrum policy using the policy language produced by the language designers. The policy administrator makes the policy available to spectrum users. Administrators do not have to know all the details of the language presented here, as they will likely encode the policy by using a (perhaps graphical) tool, called an instance editor, which hides the notational complexity of the language. The administrator also uses analysis tools to verify in advance that the encoded policy will have the desired effect and is consistent with existing policies.

In the short term, policy may be described first in English using engineering and legal terms that are commonly understood by the community of interest. Policy administrators will likely continue to use established procedures to interpret spectrum policy prior to encoding.

Given a policy expressed in English, the first step is to perform an analysis to determine how to structure the policy in a form best suited for machine understanding. Once this analysis is performed and the elements of the policy are mapped to the elements of the XG policy language, the policy can then be readily encoded in the XG policy language (XGPL). BBN has developed conversion tools that enable the use of a shorthand notation. Other tools to validate the syntactic and logical correctness of the policy represented are the subjects of future work. As this technology matures, we anticipate that a number of graphical and scripting tools will be developed to make the process easier.

A *spectrum user*, such as an XG system, must be able to use the policy to assess whether policy allows identified potential spectrum opportunities and to understand the constraints the policy administrators have placed on their use. The next section details how the spectrum user uses the policy.

3.1.2 Policy Usage

In this section, we describe a concept of operations for how a policy-agile spectrum user, such as an XG radio system, can use machine-understandable policy. A radio can use the machine-understandable policies to assure that its use of spectrum conforms to policy, as well as to modify radio behaviors in order to identify and utilize spectrum opportunities that are authorized. In this section, we will focus on the former, namely, how to assure policy conformance. The draft XG Abstract Behaviors RFC [XGAB] describes XG radio behaviors related to acquiring and sharing spectrum awareness, as well as behaviors related to identifying, selecting, coordinating, and using available spectrum opportunities as authorized by policy.

Figure 5 is a logical functional block diagram of the policy usage concept of operations. The placement of these functional blocks in hardware or software is to be determined by the designers of each individual system.

There are four main components in this functional decomposition:

- ***Sensor:*** provides situational information to the radio about the spectral environment at a given location and time. This information is key to being situationally aware of spectrum opportunities enabled by policy. We note that the sensor outputs need not be limited to the RF spectral environment; the outputs could include a variety of other information (e.g. geo-location, temperature, and proximity to specific targets) that could be used as parameters for system policy.
- ***Radio Platform:*** provides the basic hardware and primitives of a host radio system that enables opportunistic use of spectrum (e.g., RF front end, DSP hardware, system software including the OS, middleware, and libraries, and primitives for networking protocols, waveform agility, and beam forming).
- ***Policy Conformance Reasoner:*** manages accredited policy information, which includes interpreting the policy language, and reasoning based on accredited policy (e.g., approved by a regulatory authority) and related background knowledge to determine whether the proposed spectrum use is policy-conformant or not; this key component to ensuring policy conformance is largely system independent and does not tell the radio platform or the system strategy reasoner what to do.
- ***System Strategy Reasoner:*** determines the system's strategy for opportunistic spectrum sharing under regulatory and system policy constraints; this reasoner is aware of system-

specific optimizations and tradeoffs and has control over the radio platform. In Figure 5 we show the system strategy reasoner as logically separated from the rest of the radio. This highlights the potential for reusability.

We envision that the radio platform includes an accredited kernel, within which it implements (grounds) accredited parameters and processes governed by policy. We further envision that only the policy conformance reasoner, the sensor, and the accredited kernel of the radio platform are within the accreditation boundary (shaded portions in Figure 5). Parts of the system that lie within the accreditation boundary are both necessary and sufficient to ensure policy conformance. As per the XG vision [XGV], system and protocol innovations including the system strategy reasoner are outside the accreditation boundary.

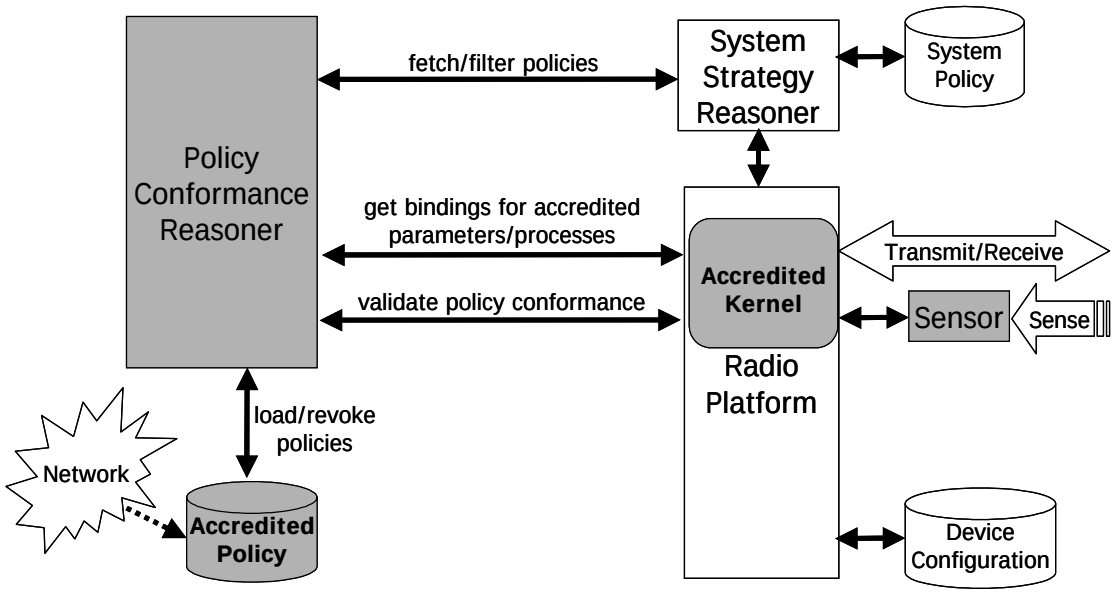


Figure 5: Policy-Agile Operation of XG Spectrum-Agile Radio

This architectural separation of the regulatory policy conformance function from the system dependent optimizations and tradeoffs allows the policy conformance reasoner to be reused and eases accreditation. Rather than separately accredit each of n radio configurations for each of m policy sets for a total of m × n operations, this approach reduces the required number of operations to m+n+1. The policy conformance reasoner must be accredited once, each policy set once, and each radio configuration once.

We describe the functions and interactions of the policy conformance reasoner in more detail here. The policy conformance reasoner performs three basic services:

- First, it loads (and possibly revokes) and interprets accredited policy instances, such as policies specified by a regulatory body, which are represented using the machine-understandable policy language.
- Second, it makes use of the policy structure to respond to queries (by a system strategy reasoner) to filter policies based on selection criteria (e.g. to a specified radio, or intended operating environment).
- Third, it determines whether or not the spectrum use proposed by the radio platform conforms to policy.

In order to explain the operation of the conformance reasoner, we provide a brief overview of the structure of policy rules in the XG policy language. Each policy rule is composed of three parts: a

selector description, an opportunity description, and a usage constraint description. A *description* is a Boolean predicate expression that involves regulated parameters and methods (for which bindings and groundings are provided by the radio platform) and additional policy parameters (for which bindings are provided within the policy itself). An *instance* is a set of bindings that can be used to determine the truth-value of a *description*.

A policy is *selected* and applied if the proposed selector instance satisfies the selector description. If the proposed opportunity instance matches the corresponding opportunity description, this provides *authorization* for the opportunity[2]. The proposed usage instance must fulfill certain *obligations*, i.e., it must satisfy the corresponding usage constraint descriptions. The use of selection, authorization, and obligation to structure policy rules is a paradigm that is used in other policy systems as well (e.g., KAoS).

We now present a notional sequence of operations using the functional decomposition presented in Figure 5. After loading any configuration information, the radio platform in conjunction with the sensor (potentially using a sensing strategy determined by system strategy reasoner) acquires awareness of its situation. The system strategy reasoner has access to the configuration, state, and awareness acquired by the radio platform through means that are specific to the system implementation. Based on system and regulatory policy and knowledge of the radio platform, the system strategy reasoner enables the radio platform to identify and characterize available opportunities and a suitable use of those opportunities. The characterization can result in, for example, binding values to relevant parameters.

Once an opportunity is identified and its intended use is characterized as a set of selector, opportunity, and usage constraint instances, the radio platform must present this set to the policy conformance reasoner for validation. In order to validate the instances, the policy conformance reasoner may need to obtain parameter bindings and invoke method groundings implemented within the accredited kernel of the radio platform. If the requested set of instances is validated, the XG radio may then use them to transmit.

The system strategy reasoner has the function of influencing radio behaviors in response to policy (and situational knowledge) in order to identify and utilize available spectrum as authorized. We note that there is significant scope for design diversity, innovation, and optimization within the system strategy reasoner, as well a potential for its reuse across several radio platforms. A simple system strategy reasoner may try only a limited range of opportunities and uses known to work with the radio and typical policies. A more sophisticated system strategy reasoner can include dynamic constraint solving capabilities; for example, it can query the policy conformance reasoner for applicable policy constraints (based on the state of the environment and the capabilities of the radio platform), and then determine a strategy to traverse the policy decision space efficiently to find good opportunities for the radio platform to use.

The design and specification of the interfaces between the radio platform and the policy conformance reasoner is a subject for future work.

3.2 Language Overview

This section provides a high-level overview of the XG policy language (XGPL) including the language elements, and the policy processing logic. In this section we also present the requirements that motivate the use of OWL as the machine-understandable representation for the XG policy language.

[2]Positive and negative authorizations are supported.

3.2.1 Language Elements

An XG policy instance consists of a set of facts and rules. Facts define the policy and rules describe how to process the facts and hence how the facts relate to one another.

A policy rule is a statement of policy consisting of a set of facts, but not the rules for interpreting those facts. All other facts either further explain a policy rule or make statements about a policy rule. Policy rules encode statements of policy, such as, “if peak received power is less than -80dBm then maximum EIRP is 10mW.” Most other facts either support the policy rules or refer to the policy rules. A policy rule links three facts: a selector description, an opportunity description, and a usage constraint description to describe a single statement in a policy, as illustrated in Figure 6.

The first fact in a policy rule is a selector description. This fact is used to filter policy rules to the set of rules that may apply to a given situation. The selector description points to one or more facts describing the authority that has jurisdiction over the policy, the frequency, time, and region the policy covers, and a description of the radio to which the policy rule applies. For example, a selector description may include filters such as “applies to operation in England” or “applies to operations in the broadcast bands”.

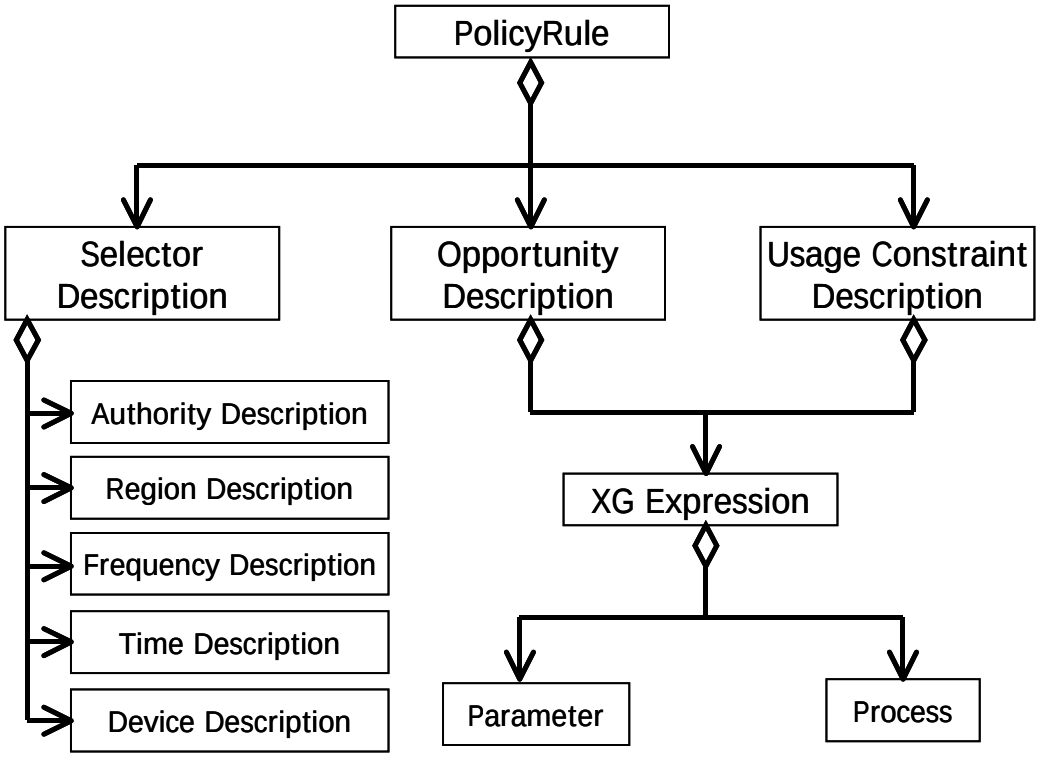


Figure 6: Structure of Policy Facts

The second fact in a policy rule is an opportunity description. This fact provides an expression that is used to evaluate whether or not a given environment and device state represents an opportunity for this policy rule. For example, it can express opportunities such as: “peak received power is less than -80dBm” or “if a beacon is heard at 823MHz.” The opportunity description is not evaluated unless the selector description from its policy rule matches. If a potential opportunity, called an opportunity instance, matches the opportunity description, a flag in the policy rule indicates if this is a valid or invalid opportunity. A valid opportunity indicates transmission that conforms to the usage constraint description is permitted. An invalid opportunity indicates that transmission is not permitted.

The final fact in a policy rule is a usage constraint description. This fact provides an expression that constrains the radio behavior, such as the emissions permitted and the corresponding sensing requirements when using the opportunity described in the policy rule. For example, it can express a usage such as: “transmit with a maximum EIRP of 10mW” or “maximum continuous on time must be 1 second and the minimum off time must be 100ms”

3.2.2 Extensible Semantics of Policy Parameters and Processes

As new radio platforms and policy sets are developed, they will have capabilities that were not envisioned when the policy language was developed. However, we will need to represent facts about these radios to fully describe selector, opportunity, and usage constraint descriptions that constitute policy. The policy language supports two generic constructs, namely, parameters and processes to represent policy concepts.

Parameter facts define all the values that are in the policy. They include values such as frequency bands, power levels, times and geographic areas. Parameters may be bound if the value is known, or unbound if the value is not yet known, such as a value that the radio platform is expected to bind.
Some policy rules will require the radio to perform certain functions to provide the information necessary to match an opportunity or usage constraint description. These are described using *process facts*. The process fact describes inputs and output parameters for the function (analogous to a function prototype in a programming language such as C) and possibly expressions constraining the relationship between the inputs and outputs. The radio is responsible for providing an implementation of the process. If a radio doesn't support a needed process, then it cannot use the opportunity that the policy rule describes.

In addition to these generic constructs, we have organized some of the key concepts in this domain into *ontologies* to provide a usable foundation. These ontologies include a structure of authority and delegation, frequency classifications, geopolitical regions and spatial descriptions, time, device capability descriptions, environment and device state descriptions, physical quantities, and units.
For example, an authority is an entity that has jurisdiction over some frequency, region, time, and set of devices and is authorized to create policy for that jurisdiction. An authority may be, for example, a regulatory agency or a primary user who is authorized to lease their spectrum to other users. A fact defines the authority and states its jurisdiction.

Some policy rules may apply only to a radio with a specific set of capabilities, either because the policy rule is designed for a specific type of radio or because it requires a particular set of parameter types or processes for the policy rule to be evaluated (e.g., supports geo-location, implements database access function, or maintains a history of parameter values). Such information is captured in a device description fact.

3.2.3 Meta-Policies

In absence of any other information, the set unions of the usage constraint descriptions from all the policy rules that represent valid opportunities apply. However, meta-policy facts may state relationships between policy rules that modify this logic. We have included three types of meta-policy facts: grouping, precedence, and disjunction.

Grouping simply creates a named set of policies so they can be referenced as a group. The group may be described either by explicitly listing the member policy rules or by creating an expression that describes the policy rules that are members of the group.

Precedence provides information on how to interpret conflicting policy rules. Two policy rules that have matching selector descriptions may also have opportunity descriptions that match a particular

opportunity instance. If both these opportunities represent a valid opportunity the usage constraints for both policy rules apply. However, if the policy rules disagree on whether the opportunity instance is valid, there is a conflict and the rule with the higher precedence is applied. Precedence may be defined between two policy rules or two policy groups. If one policy group has higher precedence than another, then all its member policy rules have a higher precedence than the member rules of the other group.

Finally, two policy groups may be disjunctive. If policy groups are disjunctive, then the policy rules in one or more of those otherwise applicable groups can be selected for application.

3.2.4 Policy Processing Logic

Policy processing, within the policy conformance reasoner for example, requires a set of rules describing how to process and interpret the policy facts. These rules govern the selection of policy rules that match a given selector instance; the selection of a policy rule that represents a valid opportunity given a selector instance and an opportunity instance; and the conformance of a given selector, opportunity, and usage constraint instance to the policy set.

The processing rules allow regulators to create policy rules that build on existing rules. Unless modified by a meta-policy, the union of all usage constraints from policy rules that represent valid opportunities is required for transmission to occur in that opportunity. Therefore, policy administrators do not have to enumerate policies for each band but may, instead, create broad policies and refine them where necessary.

For example, if a policy administrator wanted to constrain emission power in the television broadcast bands to –10dBm where an opportunity was detected, but additionally restrict emissions to have a 1MHz bandwidth in the frequencies[3] represented by channels 58-63, only two policy rules need to be written. The first rule would set the –10dBm constraint on all television channels and the second rule would set the bandwidth constraint on channels 58-63. If a radio wanted to transmit on channel 61, it would have to conform to both emission constraints, as both policy rules apply. At a later stage, regulators may easily add a third policy rule that constrains new radios of a particular type from transmitting on channel 59 in selected regions, without modifying the other two rules.

We believe that the union of simpler rules (logical implications) scales well and offers flexible management of policies. In contrast, the use of procedural structures such as if-then-else conditions and while-loops can result in brittle policy structures over time as policy evolves. For example, a small change in policy can result in a large number of changes that must be reflected across several if-then-else branches or require the entire policy to be rewritten. Furthermore, an if-then-else structure can artificially constrain the order in which conditions are tested by a particular radio.

3.3 Policy Language Representation Requirements

A standard representation for the XG policy language is necessary so that regulators can encode policies in one language and all XG radios understand the encoded policy. This section discusses the features required for representing XGPL, provides an overview for some representations considered and introduces the OWL Web Ontology Language, the standard representation used for XGPL.
Spectrum policies have a complex structure with many dimensions and layers of exceptions that are difficult even for human interpretation. In selecting a language, one must ensure that the language is capable of capturing and potentially simplifying a number of aspects of this complex structure, including:

[3] Supporting information such as the mapping of channel numbers to frequencies must exist elsewhere in the policy.

- *Inheritance*. Spectrum policy is very large and complex. The property of inheritance helps manage this complexity by enabling policy rules and properties to be extended and reduce the need for re-enumeration. For instance, rules for the 2.4GHz unlicensed band can inherit and extend rules for general unlicensed use.
- *Reification (rules about rules).* Policy rules may make statements about other policy rules. For example, we can make a policy rule governing when or where a set of policies will apply.
- *Inference (derivable rules).* There are rules that may not be explicitly stated, but follow from two or more rules. For instance, given that TV channels are 6 MHz wide, and that a policy exists to permit XG devices to transmit within a locally unallocated TV channel, one may infer that the maximum bandwidth for an XG device using an unallocated TV channel is 6 MHz. More common languages such as XML and IDL, while perhaps more straightforward for humans, do not facilitate such inference.
- *Extensibility*. It is imperative that the policy language be extensible in its vocabulary, structure, and semantics so that the language can adapt to express new types of policies as spectrum policy requirements change. Additionally, different countries may be concerned with different types of policies or different signal parameters, and it is crucial that the language be extensible to include their requirements.
- *Maintainability*. Policy will evolve over time. Small changes in policy should require only small, localized changes in the policy encoding and should not have ripple effects throughout the policy. Additionally, the language should make it easy to select policies that apply to a particular situation (i.e. time, location, frequency bands, device type). This argues for a declarative approach based on facts and rules instead of a procedurally based language as exceptions. We discuss this more below.
- *Scalability*. The policy language must scale to a wide range of devices that may have very different levels of resources available. The policy specification must be amenable to processing on devices with small memory and computational resources.
- *Standards*. Using a standard language representation is preferred as it enables reuse of libraries, applications, and tools previously developed. These tools reduce the cost of developing and using the language and increase interoperability between different tools. It is also expected that using a standard representation will facilitate the international acceptance of XGPL.

The language should be a declarative language based on facts and rules instead of a procedural language. One reason is maintainability as mentioned above. In a procedural language, policy would become a nested set of exceptions–many layers of `if-then-else` clauses. This creates two problems. First a small change in the policy may then affect many exception clauses, making it difficult to update policy. Second, the order in which exceptions are defined necessarily optimizes the policy for particular parameters. If a radio is concerned with other parameters, it will have a less optimal search.

Additionally, regulatory policy does not tell the radio what to do; it only defines what constitutes authorized use of the spectrum. However, enforcing a policy may require the results of a function that may be implemented in the XG radio. Using a declarative language, the policy can describe the function and specify rules based on its inputs and outputs without specifying the particular implementation of the function.

3.3.1 OWL

The World Wide Web Consortium's Web Ontology Language is a machine-understandable, semantic markup language. It is an extension of the Extensible Markup Language (XML) and the Resource Description Framework (RDF). OWL is based on DAML+OIL, a combination of efforts in the US (DAML–the DARPA Agent Markup Language) and the European Union (OIL - the Ontology Inference Layer) to create a machine-understandable semantic markup language. OWL provides a rich language for representing an ontology[4], that is knowledge about the interrelationship of objects, in a manner that allows a machine to make inferences.

There are three variations of OWL. OWL DL provides the maximum expressiveness of OWL while providing well-understood complexity characteristics of description logics. OWL Lite is a subset of OWL DL that trades off expressiveness for processing complexity. For example, it restricts some of the constraints available in OWL DL, such as limiting cardinality restrictions to be 0 or 1, or the restriction of property constraints locally to a class. OWL Full supports the same language constructs as OWL DL, but relaxes some of OWL DL's restrictions that provide its computational guarantees. XGPL will use OWL DL as the computational guarantees are required and it needs more expressiveness than OWL Lite provides.

3.3.2 Why OWL for XG?

OWL provides a crucial set of features that match the requirements for representing the complex structure of spectrum policy. Specifically, OWL is extensible, and it supports reification, inference, and several object-oriented features such as multiple inheritance.

Powerful tools are being developed to process OWL that build upon earlier research in the knowledge representation, logic, and theorem proving communities in order to support the semantic web.
We chose to focus on markup languages for several reasons. First, the automatic conversion from a highly structured markup format to other non-markup formats is relatively easy; the reverse transformation is typically more difficult and non-uniform. Markup languages and tools to process them are widely adopted around the world and are continually evolving, as is evidenced by the Web. These Web markup standards satisfy the need for a representation that is suitable for cross-platform information exchange and processing across nations and organizations.

In an earlier work, Mitola and Maguire investigated a wide variety of languages to represent policy for cognitive radios. They concluded, as we do, that a knowledge representation approach is the most suitable for radio policy. They recommended the development of a new language based on the Knowledge Query and Manipulation Language (KQML), a language that has limited exposure outside the knowledge-based systems community. Since the time of their study, the World Wide Web Consortium has developed the OWL Web Ontology Language. Using a language that is a W3 Consortium Recommendation offers potentially wider adoption.

OWL's predecessor, DAML, has also been demonstrated to be an effective technology in other complex policy domains. The KAoS Domain and Policy Services project has successfully used DAML to represent distributed logistics policies for the DARPA UltraLog program.

In summary, OWL provides the features required to implement a machine-understandable semantic model. This enables the building of generic applications that depend on knowledge of the semantics of the content, including generic theorem provers and reasoning engines that enable deductive

[4] An "ontology" is the knowledge about the relationships between objects. However, OWL also uses the term "ontology" to refer to an OWL file that encodes ontological information.

inference. OWL has the ability to represent the syntax of spectrum policy and the complex knowledge embedded in it. Additionally, OWL provides the ability to deduce policy constraints that are not explicitly stated for each specific case, but rather inferred from more general statements. Finally, we note that OWL is a Recommendation of the World Wide Web Consortium. Thus, we believe that OWL is an excellent long-term solution for the XG policy language.

Therefore, we provide an OWL representation of the XG policy language ontologies. We further show how to use these ontologies to specify machine-understandable spectrum policy separately from the implementation of the policy within particular radios.

3.4 Policy Ontologies

A number of ontologies that represent knowledge related to XG policy have been developed and represented using OWL. These ontologies have been made publicly available through a website and are also distributed along with the policy conformance reasoner software [PCR]. The concepts in these ontologies provide the foundation for expressing machine understandable policies, and additional concepts can be added to these ontologies to enhance the range of policies that can be supported.

We provide a summary of the ontologies here. Additional details on the content of these ontologies can be found in the XG Policy Language RFC [XGPLF].

- **Policy Language:** The "xgpl.owl" ontology specifies the central concepts for the XG policy language including classes and properties to represent policy specifications and how to group them. Policy specifications have a selector (specifying authority, frequency, region, time, and device class the policy pertains to), an opportunity description (an expression specifying the condition for an opportunity to exist), and a usage constraint description (an expression specifying constraints on the use of the opportunity).
- **Rules and Expressions:** The "xgpl-rl.owl" ontology describes an expression language for specifying opportunity descriptions and usage constraint descriptions, and a rule language to express policy processing logic. OWL semantics does not natively include rules and expressions, and the World Wide Web Consortium (W3C) is currently developing additional standards such as the Semantic Web Rule Language to address this issue. Future work should attempt to harmonize this ontology with such standards after they become available.
- **Parameters:** The "xgpl-param.owl" ontology specifies the Parameter class; device parameters governed by policy and other environmental parameters specified by policy must be individuals of this class.
- **Processes:** The "xgpl-proc.owl" ontology specifies the Process class; all functions and relations that are implemented by the device and used in policy specifications must be individuals of this class. Future work may attempt to harmonize this ontology with W3C the OWL Services ontologies being developed by the W3C.
- **Frequency:** The "xgpl-freq.owl" ontology specifies concepts related to frequency.
- **Geography and Geometry:** The "xgpl-regn.owl" ontology specifies concepts related to geographical and geometric regions.
- **Time:** The "xgpl-time.owl" ontology specifies concepts related to time.
- **Device Types and Capabilities:** The "xgpl-devc.owl" ontology specifies concepts related to device capabilities and device types.

- **Authorities and Delegation:** The "xgpl-auth.owl" ontology specifies concepts related to the structure of authority, their jurisdictions, and delegation relationships between authorities.
- **Physical Quantities:** The "xgpl-physq.owl" ontology is a utility ontology that specifies concepts related to physical quantities that are used in policy specifications.
- **Units and Conversions:** The "xgpl-physq.owl" ontology is a utility ontology that specifies concepts related to units and conversions.
- **Abstract Behaviors:** The "xgpl-abs.owl" ontology specifies concepts related to abstract behaviors (in terms of Parameter and Process individuals) that XG devices must support.
- **Environment:** The "xgpl-env.owl" ontology is a placeholder for specifying concepts related to the operational environment.
- **System-specific Extensions:** The "xgpl-sysect.owl" ontology is a placeholder for specifying concepts related to system-specific extensions.

These ontologies, currently populated with more than a hundred concepts, are able to support the encoding of a small range of notional XG policies. The above ontologies, represented using OWL, can be readily extended. Additional concepts can be imported from other standard ontologies as they become available. These ontologies must be developed further and populated with a richer set of concepts, based on inputs from domain experts – policy administrators, spectrum planners, and XG device manufacturers – in order to support a wider range of policies required for eventual deployment and maintenance.

3.5 Policy Conformance Reasoner (PCR) Software and Policy Utilities

A key component of the XG Policy Language Framework is the Policy Conformance Reasoner (PCR), which ensures that spectrum use proposed by the XG device is authorized by policy. BBN has developed a reference implementation of the PCR based on version 1.0 of the XG Policy Language Framework RFC. The source code for the PCR software and related utilities have been delivered to the participants of the XG program, and have also been publicly released with Government permission under a non-restrictive license.

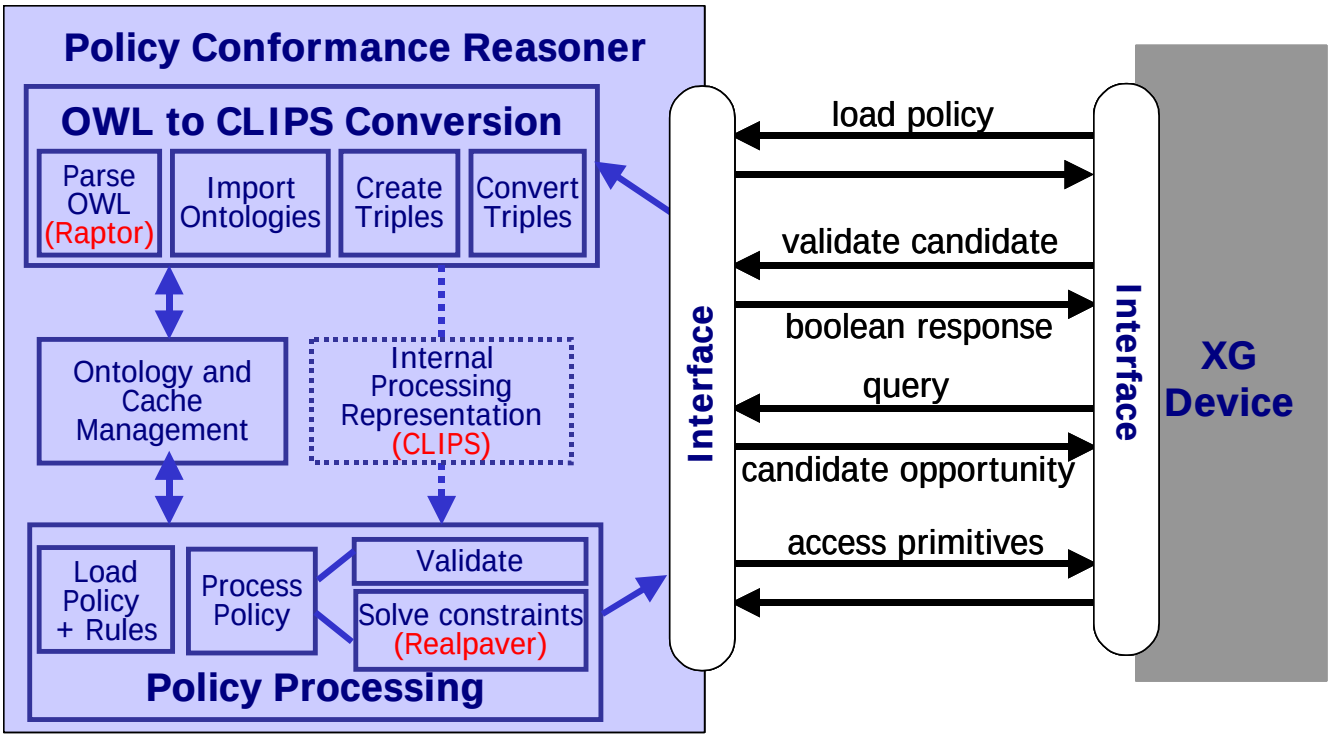


Figure 7: Policy Conformance Reasoner Software Architecture

The PCR performs the following three functions:

1. *Loads* (fetches and interprets) policy content encoded in the XG policy language that is available locally or across the network. Both the OWL representation and the shorthand representation introduced in the XG Policy Language Framework RFC are supported.
2. *Validates* the use of spectrum opportunities proposed by the XG device against the loaded policy content. The request must take the form of a fully bound opportunity, that is, all parameters that are required to validate the opportunity must be bound to values.
3. *Searches* for spectrum opportunities based on device queries that contain partially bound opportunities, and optionally device-specific constraints. Currently the search capability has been implemented as a standalone research prototype, which is to be integrated with the rest of the PCR software in follow-on work.

The PCR is based on a knowledge-based systems approach.

Loading policy content essentially populates a knowledge base with fact assertions and inference rules. Policy content may depend on background knowledge (other fact assertions and inference rules) that can be imported from existing ontologies. Many, but not all, of these facts express constraints on various parameters governing spectrum use by a radio.

Validation of proposed spectrum use involves making assertions based on what the device intends to do, applying the inference rules, and checking if the device's assertions are consistent with (i.e. do not contradict) the facts asserted by the policy. More generally, one may view this as an automated theorem-proving procedure.

Searching the knowledge base for potential opportunities that are authorized by policy involves making additional assertions on parameters that will satisfy the constraints specified by the policy. In general, one may view this as a constraint solving procedure. Several approaches are possible depending on the nature of the constraints (e.g. logical, integer arithmetic, real, linear, non-linear, or

a combination), performance requirements, and the desired solution quality. Typically no single approach works uniformly well across the range of constraints; therefore implementations must be customized to the requirements.

In order to validate and search for opportunities, the PCR requires information about the capabilities, current configuration and state of the device and its environment. For this purpose the device must provide an interface through which the PCR can access device primitives related to capabilities, configuration, and state information.

The software architecture of the PCR is illustrated in Figure 7. The PCR is implemented using the C/C++ language, and the current implementation depends on three freely available software packages (also implemented in C/C++):

1. Raptor, OWL/RDF parser libraries, which use the libxml2 libraries to parse XML and to fetch content over the Internet
2. CLIPS, a rule-based inference engine
3. Realpaver, a package for solving a range of non-linear real arithmetic constraints

The PCR interacts with the rest of the XG Device through a clearly demarcated interface, which is currently based on text strings exchanged across BSD sockets. Alternative interfaces can be created by directly accessing the lower level functions implemented by the PCR. For convenience, the interface is provided over two different sockets: one for policy management functions such as loading, and another to serve validation and search requests.

When the PCR receives a request for loading policy, it fetches and parses the policy using the Raptor libraries. Supporting ontologies that are imported by the policy and rules that encode the policy processing logic are also fetched and parsed. Sets of triples (RDF statements each containing a Subject, a Predicate, and an Object) are generated as a result of this processing. These triples are then loaded into CLIPS, and OWL processing (e.g. type inference) is done subsequently using rules we have implemented in CLIPS. Additional rules then convert the policy represented in OWL into an internal representation (quite similar to the shorthand notation) that is suitable for processing validation requests.

When the PCR receives a validation request, it asserts additional facts into CLIPS and runs the policy processing rules. If the assertions satisfy the loaded policy (i.e. there are no contradictions), then the request is validated. If the request fails to validate, diagnostic messages of conflicting facts in the policy can be output optionally.

When the PCR receives a search request, the request must include relevant selector and opportunity parameters. When making a query, the device can optionally include additional constraints of its own (e.g. opportunities with a bandwidth of at least 1 MHz) in addition to those imposed by the policy. The constraint expressions from all selected policy rules are then extracted, and output (along with any device constraints) in a form suitable for input to a constraint solver.

We have prototyped the search capability using the Realpaver software which can solve a set of non-linear real arithmetic constraints. Realpaver produces output in the form of a paving, a collection of at most *k* (a user-specified number) INNER and OUTER boxes, the union of which approximate all possible solutions that satisfy the constraints. INNER boxes represent valid opportunities usable by XG systems as is, whereas OUTER boxes must be processed further to extract valid opportunities. Alternative approaches to search and the integration of this capability with the PCR software are the subject of future work.

The PCR software can run either in the foreground or as a daemon in the background. The policy language ontologies can be fetched directly from their respective URLs, or the PCR can be

configured to use locally cached copies. A test harness called the PCR Client is included in the distribution along with an encoded policy example and sample validation requests.

In addition, format conversion utilities that enable the use of a shorthand notation (described in [XGPLF]) for representing policy have also been developed. General-purpose graphical editing environments for OWL are in their infancy. Until they mature and can be customized for editing XG policy, our conversion utilities offer a convenient text-based alternative to encoding XG policies.

Details on how to use the software and the related utilities, and a syntax specification for the PCR interface can be found in the distribution.

3.6 Encoded Policy Examples and PCR Tests

In this section, we provide an overview of the capabilities of the Policy Conformance Reasoner software through a sampling of the variety of policies it can handle, and a characterization of its performance for validation.

3.6.1 Opportunity Validation and Search using an Example Policy

The first example we describe simple policies that provide traditional opportunities to the radio without requiring sensing. In this example, Policy set E1 sets up a maximum power spectral density profile over a range of frequencies between 1.433 and 1.445 GHz. Additional constraints on the maximum continuous ON time and the minimum OFF time (between consecutive ON times) are also specified. This policy is illustrated in the top left corner of Figure 8.

We encoded this policy using the XG policy language and loaded it in the PCR. Using test scripts we emulated the behavior of an XG device by making opportunity validation requests to the PCR. The requests generated by the test scripts sampled the parameter space uniformly. The results of the validation test are illustrated in the top right corner of Figure 8 with green marks for valid opportunities and red marks for invalid opportunities. The PSD vs. Frequency plot indicates that the PCR is interpreting Policy E1 as expected.

In order to demonstrate the ability to incrementally modify policy, we add Policy E2, which modifies Policy E1 by adding a notch-out band between 1.442 and 1.443 GHz where no emissions are allowed by XG devices. The combined policy is illustrated in the bottom left corner of Figure 8. We encoded Policy E2, loaded E1 and E2 combined into the PCR and ran validation tests. The results plotted in the bottom right of Figure 8 show the presence of the notch-out band as expected.

Next we output the constraints set by the policies E1 and E2 in a form suitable for input to the realpaver constraint solver. The solver inputs and output for our example are shown in Table 1.

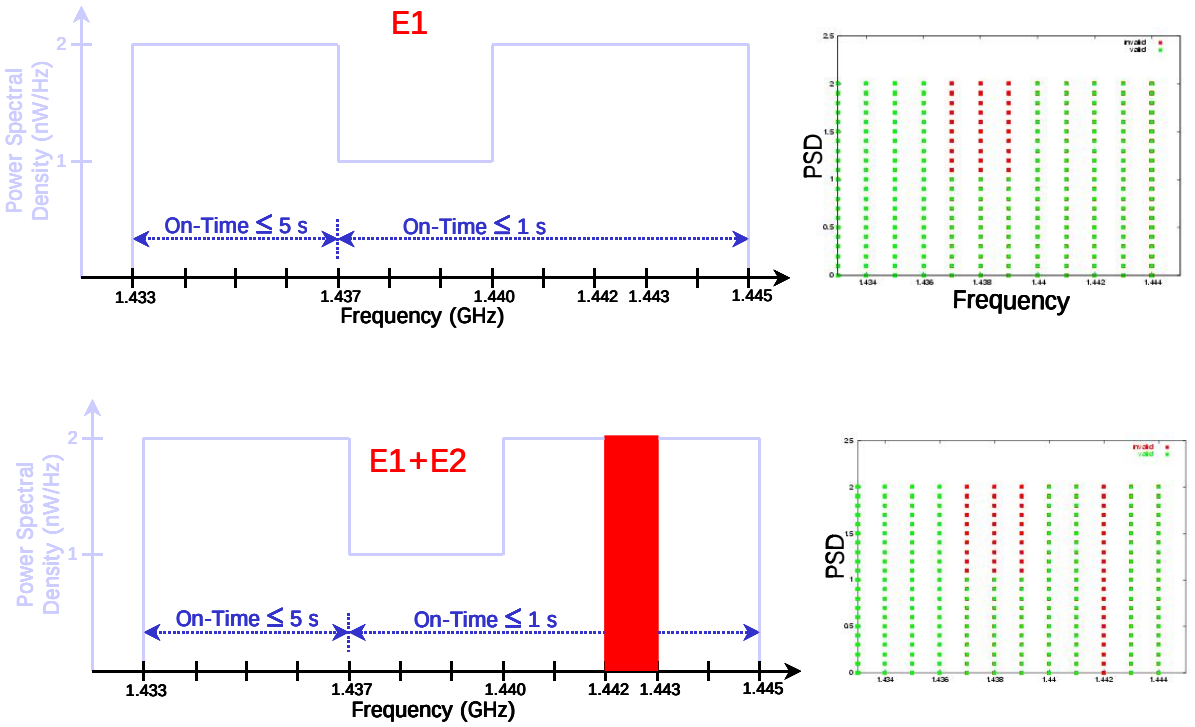


Figure 8: Policy E1 with PSD Constraints Modified by E2 to introduce a Notch-out Band

Table 1: Opportunity Search Based on Non-Linear Real Constraint Solving

Solver Input	Solver Output

Bisection choice = mn, mode = paving; /* realpaver tool directives */
Constants
f0 = 1.433E9, f1 = 1.436E9, f2 = 1.440E9,
f3 = 1.442E9, f4 = 1.443E9, f5 = 1.445E9;
Variables
XmitMin in [0.0, +oo[, XmitMax in [0.0, +oo[,
PSD in [0.0, +oo[, OnTime in [0.0, +oo[;
Constraints
XmitMax >= XmitMin, /* background info */
XmitMin >= f0, /* selector */
XmitMax <= f5, /* selector */
 	
/* conjunctions and disjunctions encoded using min and max */
min(f4 - XmitMin, XmitMax - f3) <= 0.0, /* notch-out */
min(PSD - 1.0,
min(max(max(1.433E9 - XmitMin, XmitMax - 1.436E9), PSD - 2.0),
max(max(1.440E9 - XmitMin, XmitMax - 1.445E9), PSD - 2.0))) <= 0.0,
min(OnTime - 1.0,
max(max(1.433E9 - XmitMin, XmitMax - 1.436E9), OnTime - 5.0)) <= 0.0;
	Query:
	(requestOpp)
	Response:
	INNER BOX 1
	XmitMin in [1437000000,1438111111.111111]
	XmitMax in [1438666666.666667, 1440333333.333333]
	PSD in [0, 1]
	OnTime in [0, 1]
	
	Precision: 1.67e+06, elapsed time: 10 ms

The solver returns 1024 boxes (by default) that contain all the solutions. The very first box is usable, but does not represent the best available opportunity for the radio. An issue for this approach is that when the problem in under-constrained, some solutions may be trivial or unusable by the device, and useful solutions contained within OUTER boxes require further narrowing. Although all solutions are contained within the 1024 boxes, subsets of combinations of the boxes are valid, but it is not practical to compute and test them all.

A more useful approach is to pass constraints from the device to the solver through the opportunity query. For example, in the above case, if we add a constraint that the device requires a 9MHz opportunity, a unique solution is returned.

In the XG Policy Language, usage constraints are included in the UseDesc class and these constraints can be extracted automatically. Therefore the solver can be integrated with the PCR by writing some glue software.

3.6.2 Other examples

We will briefly describe four other sets of policy examples that we have encoded and tested the PCR for its ability to load these policies and validate opportunities. In all these cases, the PCR is able to load the policies and the results of the validation indicate the policy is interpreted correctly.

3.6.2.1 Sensing-based Policies in Adjacent and Overlapping Frequency bands

The purpose of this example was two-fold. First, spectrum opportunities of interest to XG will involve the sensing of primary signals, and policies that enable such sharing are required. Second, it is desirable to be able to combine policies in frequency bands that are either adjoining or overlapping.

This example contains three sets: P1, P2, and P3. P1 applies to the frequency range 3.6-3.7 GHz, P2 applies to the adjoining frequency range 3.5-3.6GHz, and P3 applies to the frequency range 3.45-3.55 GHz, which overlaps P2.

P1 authorizes XG devices to transmit with a maximum PSD of 1nW/Hz if the incumbent signal power is sensed to be –105dBm. The corresponding values for P2 are 2nW/Hz and –80dBm. The corresponding values for P3 are 1.5nW/Hz and –95dBm.

All three policies impose additional constraints on the minimum look-through interval for sensing, maximum ON time, minimum OFF time, and maximum power leakage outside the band.

*3.6.2.2 **Geographical Policies***

The purpose of this example was to specify policies for location-aware devices. Two policies P4 and P5 were specified that applied to the same frequency range but with different operational constraints within different regions (specified by latitude and longitude ranges).

*3.6.2.3 **Time-based Policies – Opportunity Expiration***

The purpose of this example was to specify time-based policies. The ability to expire policies enables experimental policies that can then be replaced by other policies at a later time. Two policies T1 and T2 that applied to the same frequency and region were specified. T1 expired at midnight on a particular day and T2 (which relaxed transmission power constraints) took effect after T1 expired.

*3.6.2.4 **Reuse of Television Spectrum***

The purpose of this example was three-fold. First, there is interest within the industry and the regulatory community to investigate the reuse of television spectrum, and therefore it is useful to consider policy examples for this scenario. Second, the ability to specify policies that involve a functional relationship between parameters is of interest. Third, it is useful to be able to specify policies with different constraints based on varying device capabilities.

This example included two policies TV1 and TV2 that applied to television channels 60-69. TV1 applied to devices based on simple energy sensing, and authorized a maximum PSD of –53 dBm/Hz if the sensed power was below -100 dBm. TV2 applied to more capable devices that employ sub-noise detection of DTV signals, and authorized a transmit PSD of *(-160 – sensed power)* dBm/Hz provided the maximum sensed power of the DTV signal was below –107dB/m, and the maximum transmit PSD did not exceed -40 dBm/Hz.

3.6.3 Policy Tools Capabilities Summary

The capabilities of the policy tools developed in the XAP project are summarized in Table 2. With further development, we believe that the technology can be transitioned to XG systems in follow-on work.

Table 2: Policy Tools Capabilities Summary

Capability	Status
Adjacent band policies	Yes, can express, validate, and have example
Overlapping policies	Yes, can express, validate, and have example
Simple geographic constraints	Yes, can express, validate, and have example
Simple temporal constraints, expiration	Yes, can express, validate, and have example
Device capability based policies	Yes, can express, validate, and have example
Off-line merge of overlapping policies	Yes, can validate, and have example
Extensible, declarative, incrementally modifiable	Yes, based on semantic web and rule engine
Validation performance and scalability	Yes, sub-second response on PC with 10s of policies and 100s of facts, significant gains possible from straightforward implementation changes
Disjunction and other meta-policies	No, can express, but not validate, and complicates search (selectors are a practical workaround)
Candidate crosses policy region (different set of rules apply to different parts of same candidate)	No, but a practical workaround exists (use selectors, e.g. UWB specific selector), not a show-stopper
Supporting background ontologies	Yes, but quite sparse, and needs enhancement driven by further policy use cases, proposed for Phase 3
On-line merge/revoke of policies	No, but proposed for Phase 3
Policy set inheritance	No, but possible through language extension
Authority delegation and user management structure	No, but possible through language extension
Validation engine	Yes, software already available, more performance and robustness enhancements, and CORBA interface planned for Phase 3
Search engine	No, working prototype code, anticipate integration early in Phase 3
Syntax and logical consistency checker	Yes, offline based on OWL consistency checkers
Human-friendly policy encoding tools	No, have engineer-usable CLIPS-based surface notation along with tools to convert to/from OWL, but a graphical development environment friendly to spectrum-manager proposed for Phase 3
Domain-specific policy interaction analyzer	No, do not anticipate need for Phase 3, but needed eventually
Trusted policy distribution and management system	No, do not anticipate need for Phase 3, but needed eventually, may leverage trusted download for software-defined radio

3.6.4 Validation Performance

In this section we report a preliminary evaluation of the current prototype PCR software. A purpose of the evaluation was to serve as a rough guideline for current capabilities while developing the XG policy technology roadmap described in Section 7.

The results of loading various combinations of policy sets that we described earlier are shown in Figure 9. The combinations considered are

- Sensing-based policies P1 (Section 3.6.2.1)
- Sensing-based policies in adjoining and overlapping bands P1+P2+P3 (Section 3.6.2.1)
- Geographic policies P4+P5 (Section 3.6.2.2)
- Time-based policies T1+T2 (Section 3.6.2.3)

- Maximum PSD and Notch-out policies E1+E2 (Section 3.6.1)
- Re-use of television spectrum policy TV1 (Section 3.6.2.4)
- All of the above

Metrics that we have plotted in Figure 9 include the number of common processing rules (tens), the number of policies, the number of precedence meta-policies after inference (tens), the number of selected policies for the opportunity to be validated, the total number of facts (tens), the time for preprocessing the encoded policy to the internal form, and the time for validate an opportunity. The experiments were preformed on a 800MHz Pentium III laptop with 256MB of RAM.

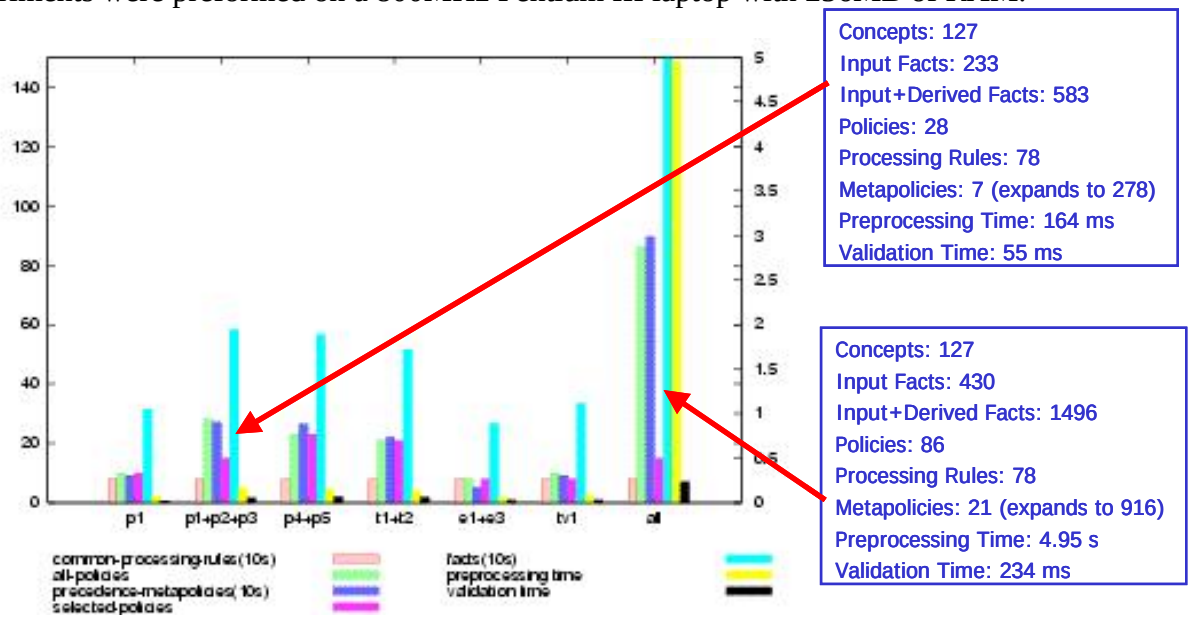


Figure 9: Validation Performance of the Policy Conformance Reasoner

The key observations we made were the following:

- PCR prototype is able to handle over a hundred concepts and over a thousand facts, covering an interesting range of policies of interest to XG, and provide a sub-second validation time response on with modest computing hardware requirements
- The current prototype loads and processes the policies upon each request. By caching the results of preprocessing the policies (e.g. processing precedence meta-policies has a quadratic complexity in the number of policies), significant performance gains can be achieved.
- It is beneficial to hierarchically organize policies so that we can limit the number of policies that are selected at any time. This will reduce the number of facts over which the inference rules must be run for each validation.

Other software enhancements that can be considered in future work are documented in the TODO list included in the PCR software distribution.

4 Abstract Behaviors

In order to facilitate the verification, validation, and accreditation process for a wide range of diverse XG systems, it is desirable to identify and formally specify the minimal set of behaviors that a system must implement to be able to safely share or use available spectrum and conform to the requirements of regulators and spectrum assignees. In short, if a device meets this minimal set of requirements, its opportunistic use of the spectrum will be "safe" from the perspective of regulators and incumbent spectrum assignees.

The challenge in specifying a minimal set of behaviors is, of course, that the range of envisioned XG systems is large. An XG system could be a simple transceiver capable of employing IEEE 802.11 signaling on any one of sixteen different frequencies, or an XG system could be a fully software-programmable radio capable of transmitting at any frequency in the range from 3KHz to 300 GHz using any one of a hundred encoding techniques. The challenge is how to specify behaviors, without requiring extraneous functionality for the simple system and without under-specifying the behavior of the fully programmable system. At the same time, the set of behaviors should be all of those necessary and sufficient for regulators (and incumbent spectrum assignees) to deem a radio safe and compliant, so that a new set of guidelines is not required for certifying each new XG implementation.

The solution to this challenge is to specify abstractions – *behaviors*, *interfaces and information objects* – that allow considerable flexibility in how the abstractions are implemented. An abstract behavior is a specification of the behavior of an XG system. Abstract behaviors interact with each other and with the system via abstract interfaces. An abstract interface is an interface that specifies a set of functions (but not their implementations), and an abstract information object specifies the information that passes across the functional interface.

In this document we specify a set of abstract behaviors that XG-compliant systems must implement; we include these behaviors in the ***accreditable kernel*** and argue that it is those behaviors that are necessary and sufficient to satisfy policy requirements and demands of spectrum assignees without requiring the development of a new set of guidelines for certifying each new XG implementation.

Related to the notion of an accreditable kernel is the notion of ***traceability*****.** An XG system implementation, therefore, must support a one-to-one mapping between the external physical behavior (e.g. emission profile) of the system and the instances of particular abstract behaviors that the system implements. Linking external behavior (which is fundamentally what spectrum regulators and incumbent assignees care about) with internal abstract behavior provides both a standard for minimalism (if it doesn't affect external behavior, it isn't a function or behavior in the minimal set) and suggests an approach to auditing radio behavior, to ensure the radio is well behaved.

Following this rationale, we have developed a set of guidelines to help determine whether a given abstract behavior must be included in the accreditable kernel or not. In order to be included, the behavior must satisfy at least one of the following:

- It is required to support opportunistic sharing of spectrum
- It is required to support policy-defined operation
- It provides traceability from policy through behavior to emissions

The abstract behaviors are only intended to provide guidelines for the high-level system design of the individual XG systems. Although the abstract behaviors will cover the important attributes that characterize XG systems, they are not intended to be detailed design- or implementation-level specifications of particular XG systems.

In the following subsection, we present an UML model of an abstract XG radio system. We present three kinds of abstractions - Interfaces, Behaviors, and Information Objects - in terms of which an abstract system can be described. We provide a high level object-oriented description of the abstract XG radio system in terms of these three kinds of abstractions. For a more detailed treatment, please refer to the draft Abstract Behaviors RFC that not only describes each of the three kinds of abstractions in significant detail, but also describes a reference radio design based on the abstractions.

4.1 An Abstract Model of the XG Radio System

In this section, we present an abstract model of an XG radio system (radio plus software on it and its interactions with the outside world). That may appear to be at odds with our goal in this document of defining a minimal set of essential behaviors, our *accreditable kernel*. The abstract behaviors that reside within the accreditable kernel, however, interact (through XG Interfaces) with other portions of the XG radio system. So in defining the accreditable kernel, we also need to present how it interacts with the portions of the radio outside the accreditable kernel. That's the focus of this section.

Since our goal is to define the accreditable kernel, and leave the details of the rest of the radio completely open to innovation (and indeed, to keep the accreditable kernel open to innovation too), this abstract model is a careful combination of concreteness and vagueness. In general, we seek to specifically describe what functionalities have to be present, yet we do not specify the details of how those functionalities are realized.

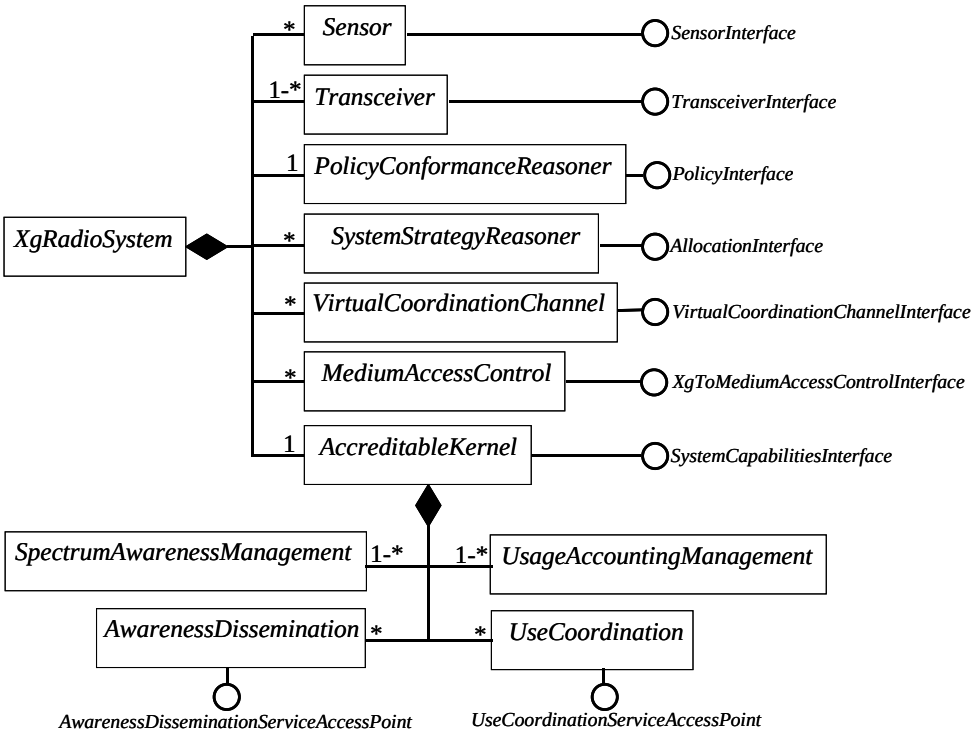


Figure 10: XG Radio System Abstract Model

In this section, we define a UML class for a radio, entitled XgRadioSystem. This class is illustrated in Figure 10 and is composed of all the components that an XG radio might need. Many components are optional, but some components must be present, and some components may be present one or more times.

The purpose of this representation is to illustrate and represent the basic hardware and primitives of a host radio system that enables opportunistic use of spectrum (e.g., RF front end, DSP hardware, system

software including the OS, middleware, and libraries, and primitives for networking protocols, waveform agility, and beam forming). This configuration of components is only one of many ways that one might choose to modularize the functions of an XG radio.

In general, each component of the XgRadioSystem class has a related interface. (It is important to keep in mind, however, that other classes may also implement a particular interface. The mapping shown in Figure 10 simply indicates that the class, if present, implements the associated interfaces). The accreditable kernel interacts with the various components through these interfaces, using information objects. So in this section we describe what each component does, and define a generic class, XG Interface, from which each component's interface will be derived. In anticipation of defining the behaviors of the accreditable kernel, we define a generic class of behaviors, XG Behavior. Finally, we define three generic classes of XG Information Objects from which all information objects used in the interfaces will be derived.

4.1.1 Subsystems of the XG Radio System

The various components of the XgRadioSystem class serve different purposes. We describe each component and its function here.

4.1.1.1 AccreditableKernel (required; one instance)

The AccreditableKernel class has the following key functions:

(i) Implement the included XG abstract behaviors and provide access to them;

(ii) Manage access to the state, configuration, and capabilities of the radio; and

(iii) Manage access to primitives implemented within the radio that can be governed by policy.

Every instance of the XgRadioSystem must have an instance of the AccreditableKernel.

The AccreditableKernel class implements an interface (the SystemCapabilitiesInterface) through which it provides access to primitives deemed necessary to ensure that control of the radio's visible behavior is located in the AccreditableKernel and that the behavior is controlled in a fashion consistent with XG policies. These primitives include Parameters and Processes described within the XG Policy Language Framework (see [XGPLF]).

The accreditable kernel also has an association with every other subsystem in the radio for each primitive implemented in that subsystem (these associations are not shown in the diagram above).

4.1.1.2 Sensor (optional; may have more than one)

The Sensor class provides situational information to the radio about the spectral environment at a given location and time. The sensor outputs need not be limited to the RF spectral environment; the provided information could include a variety of other values such as geo-location, temperature, and proximity to specific targets that could be used as parameters for system policy.

Sensors are optional because radios may learn through other means, such as configuration or through dissemination protocols, what spectrum use local operating rules authorize. Alternatively, the sensor functionality may be implemented within the transceiver subsystem. There may be more than one sensor because a radio may sense different information using different devices.

The Sensor class implements the interface called SensorInterface.

4.1.1.3 Transceiver (required; at least one)

The Transceiver class provides radio communications capabilities, including the RF front-end and the baseband signal processing functions. In general, we expect these communications capabilities to be agile and that the interface to the Transceiver will reflect this agility.

Instances of XgRadioSystem contain at least one Transceiver, by which we mean it must have at least a receiver or a transmitter capability. (Without a transceiver, we don't have a radio!).

The Transceiver implements the interface called TransceiverInterface. In cases where the transceiver is also a sensor, it may make sense for the transceiver to implement SensorInterface as well.

4.1.1.4 PolicyConformanceReasoner (required; one instance)

The PolicyConformanceReasoner class determines whether proposed spectrum use is consistent with accredited policy (e.g. approved by a relevant regulatory authority and incumbent spectrum assignee), knowledge of the local spectrum (e.g. information from the Sensor class), and other background knowledge. Note that the PolicyConformanceReasoner primarily determines if proposed use is acceptable – the task of developing proposed usages is left to other subsystems, for example, the SystemStrategyReasoner. The PolicyConformanceReasoner may, however, perform additional policy functions. It can support queries on policy to extract authorized usages for a given situation, support filtering policy information to obtain a subset of policies that apply to a current situation, generate machine proofs that a given usage conforms to policy, and manage the loading and revocation of policy sets.

Instances of XgRadioSystem must contain a PolicyConformanceReasoner, as the PolicyConformanceReasoner implements the functions that determine if the XG radio is in compliance with any policy restrictions and the abstract kernel depends on conformance reasoner to accredit usages

The PolicyConformanceReasoner class implements the interface called PolicyInterface.

4.1.1.5 SystemStrategyReasoner (optional; may have more than one)

The SystemStrategyReasoner class determines the system's strategy for opportunistic spectrum sharing given the constraints of sensor information, regulatory and system policy constraints. This reasoner is aware of system-specific optimizations and tradeoffs and has control over the radio platform.

In many ways, the SystemStrategyReasoner is the complement of the PolicyConformanceReasoner. The conformance reasoner determines whether a particular type of spectrum use is authorized by policy in the current environment; the SystemStrategyReasoner determines what opportunities to use spectrum exist in the current environment that are suitable for the XG radio. It is important to note that the strategy reasoner is not the only place in an XG radio that can identify or allocate opportunities: opportunities can be found from other nodes through an awareness dissemination protocol or this function can be incorporated within a medium access control protocol.

The SystemStrategyReasoner class, therefore, is one of several classes that can perform the opportunity allocation function, and so it implements the interface called the AllocationInterface. This class can access the state, capabilities, and configuration information through the SystemCapabilitiesInterface.

4.1.1.6 VirtualCoordinationChannel (optional; may have more than one)

In many situations, XG radios need to communicate with neighboring XG radios to coordinate awareness of the spectrum (e.g. are a set of frequencies unused at both sender and potential receivers?) and coordinate use (e.g., jointly determine which frequency bands will be used). Coordination channels can be pre-configured, discovered, or created when needed. Furthermore, the channel may be

implemented using waveform-level signaling, as a specialized MAC layer, or even at the application layer using higher layer protocols using out-of-band network access.

The VirtualCoordinationChannel class represents the logical communication channel used for this purpose.

The VirtualCoordinationChannel class implements the interface called VirtualCoordinationChannelInterface.

4.1.1.7 MediumAccessControl (optional; may have more than one)

The MediumAccessControl class includes higher layer functionality such as neighbor discovery, topology management, and link scheduling and sends/receives Layer 2 protocol data units to the transceiver. In an opportunistic spectrum-sharing environment, the higher layers need to interact with the XG radio. For example, links must be scheduled such that communicating peers select and use common opportunities that are deconflicted from opportunities used by other nodes.

The MediumAccessControl class implements the interface called XgToMediumAccessControlInterface.

4.2 XG Interfaces

In this subsection, we sketch the functions of generic Interface class and briefly mention the purpose of each of the major interfaces in XgRadioSystem.

As illustrated in Figure 11, all XG interfaces extend the Interface abstract class, or its subclass ServiceAccessPoint. In addition to the seven interfaces associated with the subsystems described in Section 4.1.1, we define two additional interfaces that are associated with the protocol behaviors included in the AccreditableKernel.

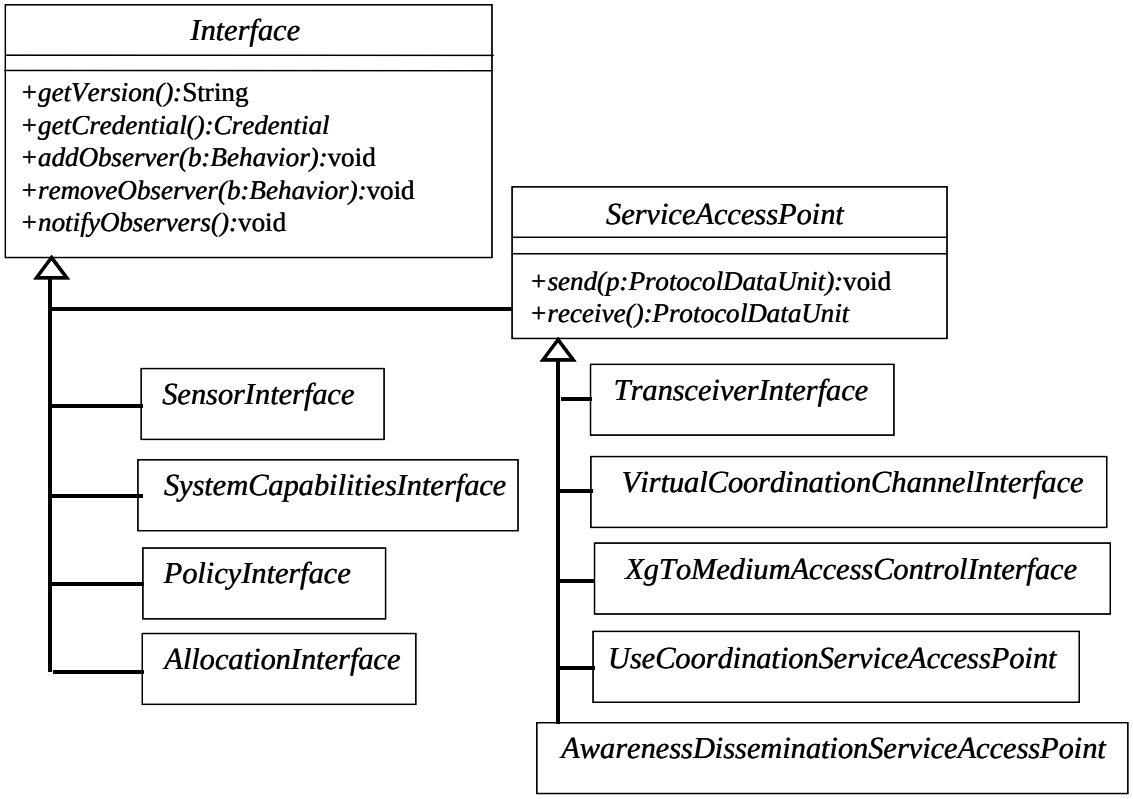


Figure 11: XG Interfaces

The Interface class provides five methods:

- *getVersion*, a method used to learn the version of the Interface; and
- *getCredential*, a method used to get credentials of the class that implements the interface, for example, digital certificates that assert that the implementation has been accredited. *getCredential* allows trust establishment mechanisms to be established to work across XG interfaces. This enables various components of the radio to be accredited and enhanced separately, and the use of particular combinations to be governed by policy if needed.
- a set of three methods that behaviors use to register and unregister at an interface and to be notified of events. Behaviors and Interfaces together implement an Observer–Subject design pattern [DP]. A behavior can register or unregister itself at an interface by calling the *addObserver* and *removeObserver* methods respectively. The *notifyObservers* method is called, which in turn calls the *update* methods on all instances of Behavior that are registered with the interface. For example, the SpectrumAwarenessManagement behavior can register with the SensorInterface and the AwarenessDisseminationServiceAccessPoint in order to know when new sensed awareness information or protocol based awareness information is received.

The ServiceAccessPoint class extends the Interface class with two additional methods:

- *send*, a method to send a protocol data unit. Note that the definition of protocol data unit (PDU) is somewhat broader in this document than is traditional – in this context, a PDU is protocol data plus a set of associated attributes which often includes information about what channel the PDU is to be sent on
- *receive*, a method to receive a protocol data unit, typically invoked by an observer in response to a notification

XG abstract behaviors interact with each other and with the system through the following extensible interface set:

1. **SensorInterface:** The interface through which sensed awareness (of the operational environment, i.e. information such as location and spectrum) is accessed and sensing behavior is controlled.
2. **TransceiverInterface:** The interface through which the agile XG transceiver (i.e. parameters such as transmit power, frequency, waveform, and beamform) is controlled and emission constraints are conveyed.
3. **SystemCapabilitiesInterface:** The interface through which the capabilities, the current configuration, and state of the XG system can be accessed.
4. **PolicyInterface:** The interface through which opportunity instances are validated against applicable regulatory, incumbent spectrum assignee, and system policy. This interface may additionally provide access to policy information and policy management directives.
5. **AllocationInterface:** The interface through which a specific opportunity is allocated for use from all available opportunities.
6. **VirtualCoordinationChannelInterface:** The interface through which virtual control channels (which carry XG protocol data) are managed and accessed.
7. **XgToMediumAccessControlInterface:** The interface through which the MAC layer can interact with the XG abstract behaviors (for example, to support link setup and maintenance, contention management, and framing).
8. **AwarenessDisseminationServiceAccessPoint:** The interface through which the AwarenessDissemination protocol behavior can be accessed. The information accessed through this interface is processed spectrum awareness that is acquired from or is to be disseminated to other nodes by the protocol (rather than the individual protocol data units exchanged by a protocol implementing this behavior over virtual coordination channels)

9. **UseCoordinationServiceAccessPoint:** The interface through which the UseCoordination protocol behavior can be accessed. The information accessed through this interface include higher level protocol directives resulting in acquisition and release of opportunities (rather than the individual protocol data units exchanged by a protocol implementing this behavior over virtual coordination channels)

XG systems that implement these interfaces are anticipated to inherit and extend these interfaces and the related classes in order to support real-time operation, access control, error or exception handling, enhanced features and system-specific optimizations.

4.3 XG Behaviors

XG systems implement the abstract behaviors necessary to enable opportunistic use of spectrum, policy-defined operation, and traceability. As a guideline to identifying which abstract behaviors are necessary, we use the following rationale. XG systems must minimally agree on how they characterize spectrum conditions (spectral awareness) and on the opportunities available to use spectrum (opportunity instances), and must support behaviors that enable the dissemination of spectral awareness and opportunity information. In addition, for traceability, they must account for emissions they make in terms of the policies that authorize each emission, and the valid opportunities that were identified to enable the emission. Finally, they must support behaviors that allow systems to coordinate the use of opportunities either to form a data communications channel (at Layers 2 and above), or to avoid opportunities selected by other XG systems.

XG abstract behaviors extend the Behavior abstract class. As illustrated in Figure 12, the subclasses of Behavior include:

- InternalBehavior, the class of behaviors that are *internal* to the XG system
- ProtocolBehavior, the class of behaviors that involve communications with agents external to an XG system. The protocol behaviors use virtual coordination channels to exchange XG protocol information, and access to the protocol behaviors can be made through a ServiceAccessPoint interface.

Behaviors interact with other behaviors and the rest of the system through Interfaces. Behaviors and Interfaces implement the Observer-Subject design pattern. A behavior can register itself at an interface, and the behavior implements a generic *update* method that can be called by the interface to notify the behavior of any state changes.

We have identified the following four XG abstract behaviors:

1. **SpectrumAwarenessManagement:** an InternalBehavior, which describes how opportunity information is acquired, identified, represented and disseminated within and across XG systems. This behavior encompasses awareness information gained from sensing, configuration, and through AwarenessDissemination instances.
2. **AwarenessDissemination:** a ProtocolBehavior, which can be used by XG systems to share opportunity awareness information.
3. **UsageAccountingManagement:** an InternalBehavior, which enables emissions to be traced to a valid opportunity enabled by policy. This behavior is responsible for ensuring that every opportunity is validated for policy conformance prior to its use. This behavior is also responsible for ensuring that all parameters governing operation are correctly bound to the values specified within the validated opportunity.
4. **UseCoordination:** a ProtocolBehavior, which allows XG systems to coordinate the use of selected opportunities with other (XG and non-XG) systems.

A natural question to ask is why these four behaviors in particular are included in the accreditable kernel and why some others (notably, allocation) not included. Recall from Section 1 that abstract behaviors must satisfy at least one of the following to be included in the accreditable kernel:

- It is required to support opportunistic sharing of spectrum
- It is required to support policy-defined operation
- It provides traceability from policy through behavior to emissions

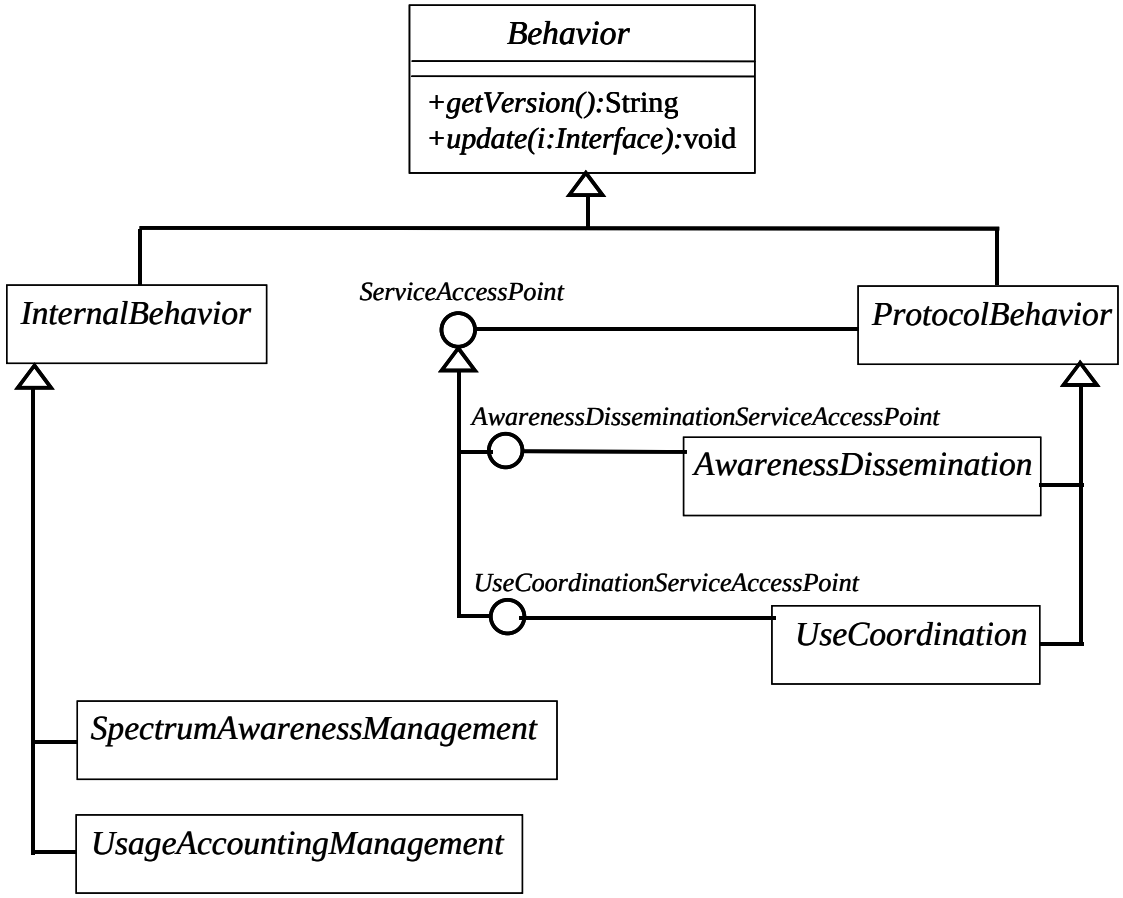


Figure 12: XG Internal and Protocol Behavior Classes

A radio without awareness of opportunities will cease to be an XG radio (i.e. it cannot perform opportunistic spectrum sharing); therefore, SpectrumAwarenessManagement is included within the accreditable kernel.

UsageAccountingManagement addresses the policy-defined operation and traceability requirements directly, and is included within the accreditable kernel.

In addition, traceability requires that trustable dissemination and coordination protocol behaviors (though not necessarily particular protocols) may be required by policy in some circumstances. An example of this requirement can be found in proposed approaches for the reuse of public safety bands through the use of beacon signals that authorize use. These two protocol behaviors (one for trusted dissemination of what's available to share and another for trusted signaling of opportunity acquisition and release) are therefore included in the accreditable kernel because they relate to policy conformance. Clearly, these behaviors are required only in those XG radios that make use of particular opportunities enabled by policy due to the implementation of these protocols.

Identification and allocation of opportunities need not be a trusted function. A tool for finding opportunities which devises wonderful spectrum use most of the time and occasionally suggests something infeasible/illegal is perfectly reasonable, provided that the infeasible opportunities are never instantiated. It is the barrier to instantiation that is the accreditable kernel's function – not the exploration of opportunities. And leaving the exploration of opportunities outside the accreditable kernel allows for freer innovation.

4.4 XG Information Objects

We organize XG Information Objects into three abstract classes as illustrated in Figure 13: PolicyDefinedJoinPoint[5], Expression, and ProtocolDataUnit.

The PolicyDefinedJoinPoint, (described further in the [XGAB]) is a generalization of two abstract classes: Parameter and Process. Instances of the PolicyDefinedJoinPoint class are primitives that are described using the XG Policy Language Framework (see [XGPLF]), and implemented within the XG radio system. The AccreditableKernel through the XG System Capabilities Interface provides access to these primitives.

Instances of the Parameter class are parameters that govern the operation of the XG radio system, and instances of the Process class are methods, procedures, or relations that are implemented by the XG radio system.

The ParameterCollection is an abstract (and arbitrary) collection of Parameter instances. Subclasses of ParameterCollection include Credential, Awareness, UsageRequest, and OpportunityInstance.
Instances of the Credential class encapsulate security and trust management information such as cryptographic keys and authentication information. Credential instances are exchanged across XG interfaces for trust management. The specification of the security and trust management architecture is not within the scope of this document.

The Awareness class encapsulates situational awareness information including spectral awareness, network awareness, and possibly temporal and geo-spatial information as well. Instances of the Awareness class are created and managed by the XG Spectrum Awareness Management behavior. Subclasses of the Awareness class include SensedAwareness that encapsulates awareness gained by the XG system through sensing, and ProtocolBasedAwareness, which encapsulates awareness acquired by the XG system through the use of awareness dissemination protocols.

The UsageRequest class and OpportunityInstance class respectively encapsulate information in the request for, and in the characterization of, spectrum use. OpportunityInstanceCollection is an abstract collection of instances of the OpportunityInstance class. Instances of UsageRequest can be sent (to the SystemStrategyReasoner, for example) over the AllocationInterface, and an OpportunityInstanceCollection that includes a list of opportunities may be returned. The radio system can choose opportunities from this collection, and then pass the selected instance of OpportunityInstance over the PolicyInterface to have it validated (i.e. checked whether it is authorized by applicable policy). A ValidOpportunityInstance object encapsulates validation status and associated credential information of an OpportunityInstance object that it contains. The XG Usage Accounting Management behavior ensures that operation (e.g. emission) conforms to the parameter values set within the ValidOpportunityInstance with the appropriate status and credentials returned by the PolicyInterface.

[5] In computer science, a *join-point* is a point in the flow of a program. In Aspect Oriented Programming, a *pointcut* is a set of join-points. Whenever program execution reaches one of the join points defined in the *pointcut*, a piece of code (called advice) associated with the pointcut is executed. We envision a PolicyDefinedJoinPoint to be points within the XG radio where behavior governed by policy is accessible.

The Expression abstract class encapsulates predicate expressions that involve PolicyDefinedJoinPoint instances. The expression language can be specific to the particular XG implementation. Subclasses of the Expression class include the PolicyFilterSpecification class, which encapsulates expressions that specify a filter on policy instances based on arbitrary criteria of interest, and the PolicyConstraints class, which encapsulates expressions that specify constraints that apply to the XG radio system. These expressions can be, for example, used by the SystemStrategyReasoner to interact with the PolicyConformanceReasoner.

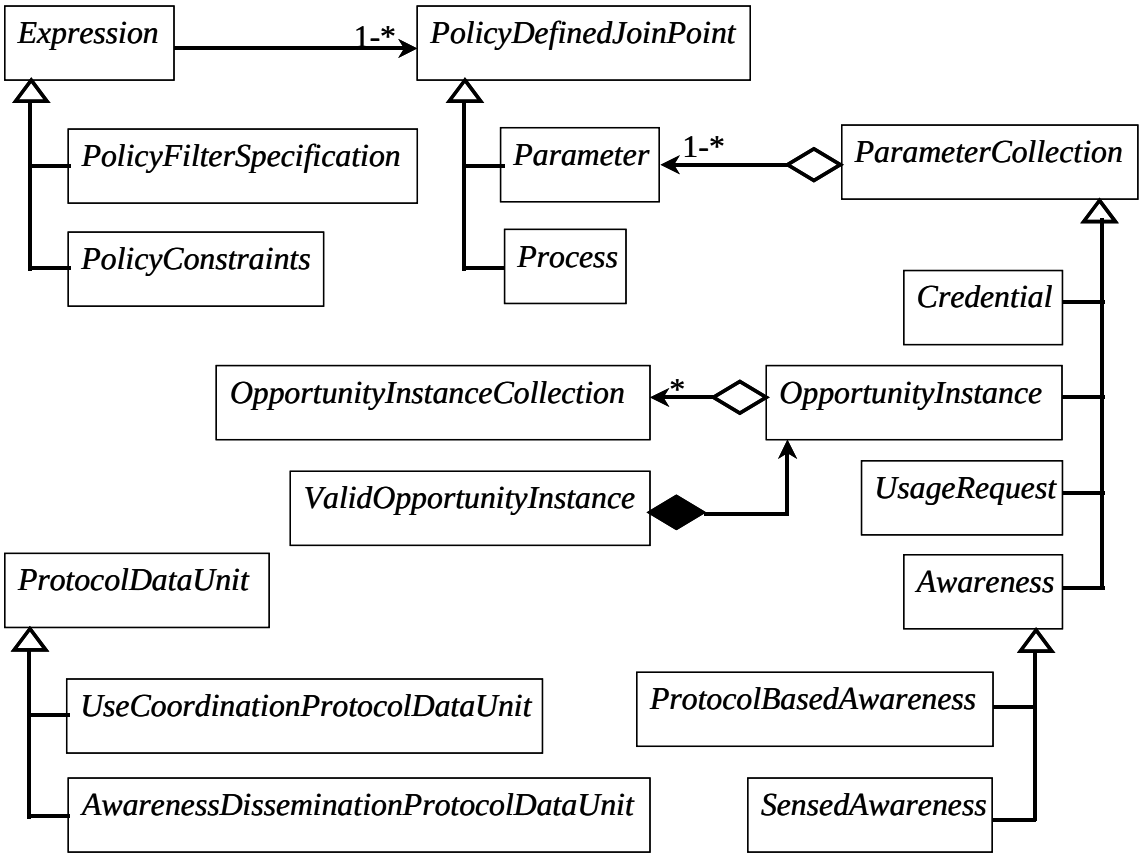


Figure 13: XG Information Objects and Their Associations

The ProtocolDataUnit is an abstract class that encapsulates information exchanged across ServiceAccessPoint interfaces (and any attributes of that information, required to effect an exchange). In particular, two subclasses of ProtocolDataUnit, namely, the AwarenessDisseminationProtocolDataUnit and UseCoordinationProtocolDataUnit are exchanged over the VirtualCoordinationChannelInterface by instances of the XG protocol behaviors (AwarenessDissemination and UseCoordination). The extension and implementation of these abstract classes are specific to the protocols used by the particular XG radio system.

4.5 XG System Interactions

There are three key system interactions that relate to opportunity awareness, allocation, and use.

The SpectrumAwarenessManagement behavior creates and manages Awareness objects, objects that provide information about the environment around the XG radio. To get the information necessary to create, update, modify, or delete Awareness objects, the behavior acquires SensedAwareness objects via the SensorInterface and ProtocolBasedAwareness objects from the AwarenessDisseminationServiceAccessPoint interface. The latter interface is implemented by the

AwarenessDissemination behavior, which exchanges spectrum information (abstractly modeled as AwarenessDisseminationProtocolDataUnit instances) with other systems via communications channels modeled by the VirtualCoordinationChannelInterface. These interactions are illustrated in Figure 14.

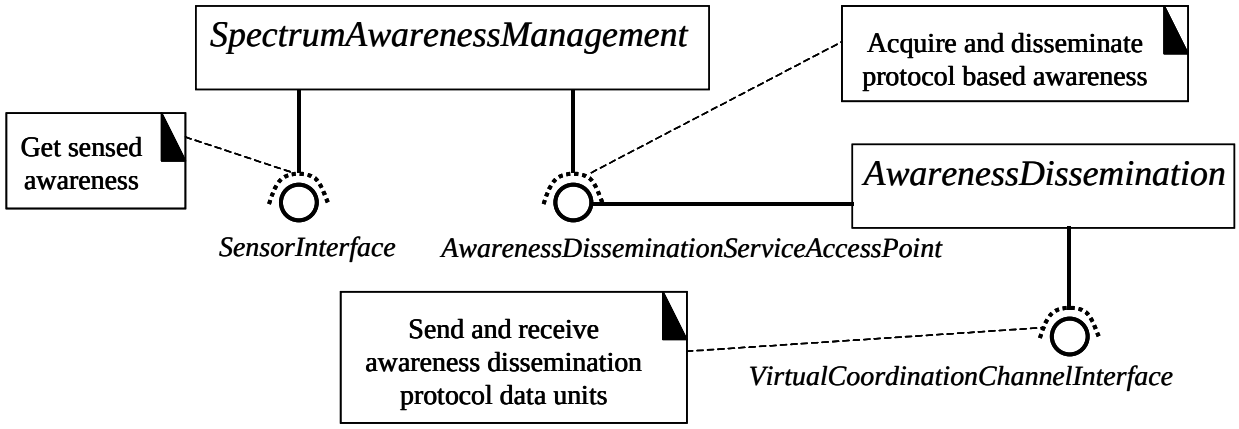


Figure 14: Spectrum awareness is gained through sensing and protocol behaviors

Once awareness of opportunities is acquired, the next step is to identify and allocate opportunities for use by the radio system under policy constraints. As illustrated in Figure 15, the SystemStrategyReasoner performs this function. The SystemStrategyReasoner can access policy information through the PolicyInterface, and it can access the capabilities, configuration, state, and primitives within the radio system through the SystemCapabilitiesInterface. In particular, this includes access to the XG SpectrumAwarenessManagement behavior.

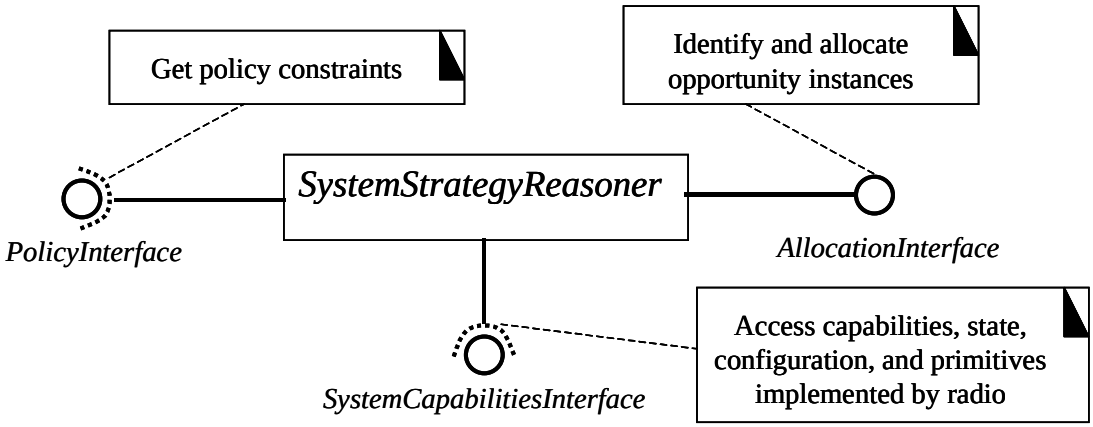


Figure 15: The System Strategy Reasoner identifies and allocates Opportunity Instances for the Radio; it may additionally access radio primitives to modify system behavior

Once opportunities have been identified and allocated based on awareness of opportunities, one or more can be selected and used in a manner that is authorized by policy. The UsageAccountingManagement behavior performs the function of ensuring that OpportunityInstance objects are presented to the PolicyInterface prior to use, and that only ValidOpportunityInstance objects returned by the PolicyInterface with proper validation status and credentials are actually used.

The UsageAccountingManagement behavior through the SystemCapabilitiesInterface asserts that subsystems actually use the parameter values contained within the ValidOpportunityInstance, before providing it to the TransceiverInterface. Furthermore, authorized use of spectrum may entail the employment of a protocol to coordinate with other nodes. For this purpose, the UsageAccountingManagement behavior interacts with the UseCoordination protocol behavior through the UseCoordinationServiceAccessPoint.

The interactions related to the use of opportunities are illustrated in Figure 16.

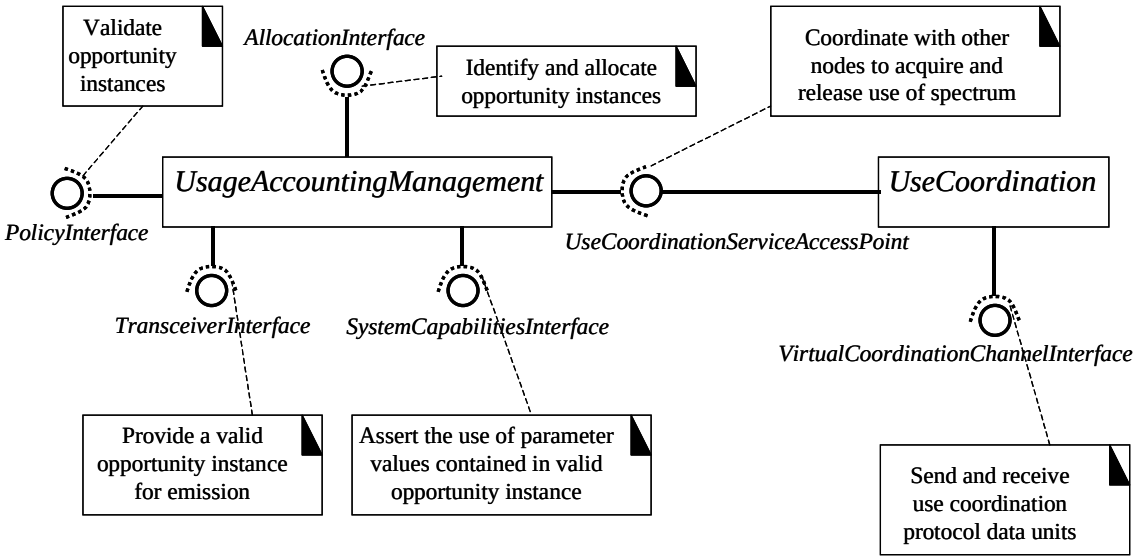


Figure 16: Opportunity instances are allocated, validated against policy, then used after asserting that all parameter values are set appropriately and after coordination with peers

A design sketch for a reference implementation of an XG system based on these abstractions is provided in the draft XG Abstract Behaviors RFC [XGAB].

5 Modeling and Simulation of Protocols for XG

During the XAP effort, we have refined and enhanced the XG Evaluation Platform (originally developed during the earlier XMAC effort). This software, developed in the OPNET environment, along with documentation, has been made available to all XG performers. The XG models and simulations developed by BBN in the context of this platform have also served the role of a proxy XG system for developing the XG framework, the abstract behaviors in particular.

5.1 XG Evaluation Platform (XEP)

XEP is an interacting set of modular mechanisms for opportunistic spectrum access. It consists of several new mechanisms working cohesively to implement a complete opportunistic spectrum access (OSA) system. In designing XEP, our objectives were:

- Backward compatibility. To develop a complete system solution, capable of exploiting opportunistic spectrum access even if the upper layers are not aware of this capability (e.g. using legacy MACs).
- Simplicity. The goal was a straightforward realization of opportunistic spectrum access that forced us to develop all the essential mechanisms and none of the optimizations. A simple system is a good benchmark against which to test optimizations, and provides a clear starting point.
- Modularity. We have sought to divide up the functionality such that adding new features or optimizations is easy, without destroying the general architecture.

Figure 17 shows the XEP system architecture. From the XEP perspective, the node is divided into three layers: physical, XEP Management, and MAC-and-above layers.

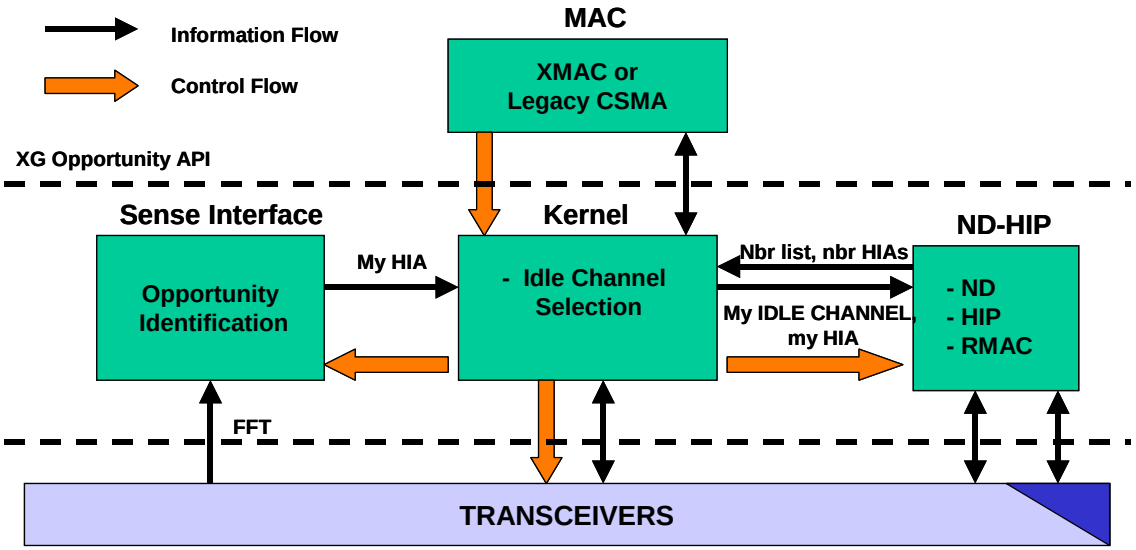


Figure 17: XG Evaluation Platform System Architecture

The physical layer consists of two transceivers. The first transceiver is frequency agile and can be tuned to different frequencies and waveforms by the XEP management layer. The second transceiver is tuned to the coordination channel. We assume that the coordination channel is allocated a priori and is known to all nodes. Each transceiver is capable of successfully decoding (receiving) one packet at a time. In addition, the physical layer includes a sensing device that periodically reports to the XEP management layer the primary nodes' energy over the frequency bands of interest.

Above the physical layer sits the XEP management layer. Among its functionalities are:

- Estimating the available frequencies in the node's neighborhood (opportunity identification).
- Discovering other XG nodes in the neighborhood and selecting those of them that are appropriate neighbors.
- Informing other XG nodes of this node's frequency observations.
- Selecting a default set of frequencies to become the node's IDLE channel. The IDLE channel is the frequency band(s) and waveform(s) that the node is tuned to when it is not transmitting, receiving, or preparing for reception of a pre-scheduled packet.
- Sharing this opportunistic node's IDLE channel identity with its neighbors so that neighbors know how to reach it.
- Serving as a front-end for the MAC layer. Beyond simply transmitting and receiving, the interface between XEP and the MAC layer allows the MAC layer to take an active role in channel selection and management.

The XEP management layer divides these functions across three modules: XEP kernel, Sensing Interface, and Neighbor Discovery and Hole Information Protocol (ND-HIP). The XEP kernel module centralizes this layer's decisions and presents a well-defined interface to the MAC layer. It also handles most physical layer functions. The Sensing Interface module is in charge of determining the presence of transmission opportunities. And the ND-HIP module is in charge of disseminating OSA information among opportunistic nodes.

In the experimental results in Section 5.3 we assumed the MAC layer was unaware of XG (e.g. a legacy MAC) and XEP was entirely responsible for finding spectrum to meet the MAC layer's needs. However, we expect that in many cases the MAC will be XG-aware and that the MAC will indicate to the XEP kernel the capacity the MAC needs.

Figure 17 also shows the information flow in XEP. The physical layer reports measured energy in various frequency bands. The Sense Interface module receives this information and converts it into a Hole Information Array (HIA). The HIA is a vector where each entry corresponds to a frequency band and indicates if the band can be opportunistically used. One should view the HIA as a joint product of the physical sensors and the Sense Interface, as the physical layer may have to be programmed by the Sense Interface to detect transmissions. The Sense Interface reports the HIA to the XEP kernel.

Concurrently, the NDHIP module exchanges HIA information with other opportunistic nodes and provides the HIA information from the node's neighbors (and perhaps peers farther away). Combining its local HIA information with that of its neighbors, the XEP kernel develops a map of how the spectrum is being used in its area, and selects the node's IDLE channel(s).

The XEP kernel provides the NDHIP module with both its HIA and IDLE channel information for the NDHIP to propagate it to other opportunistic nodes. This environment is described in greater detail in the software documentation [XEP].

5.2 Protocols for Opportunistic Spectrum Access

In this section, we describe a set of protocols for supporting spectrum agility in a Mobile Ad Hoc Network (MANET). These protocols were developed at BBN as part of the DARPA neXt Generation (XG) program, and have been implemented within an OPNET simulation model. Our main objective is to study the benefits of spectrum agility, not necessarily to design the most robust or efficient solution. We shall also briefly discuss some insights based on our experience using this simulation model.

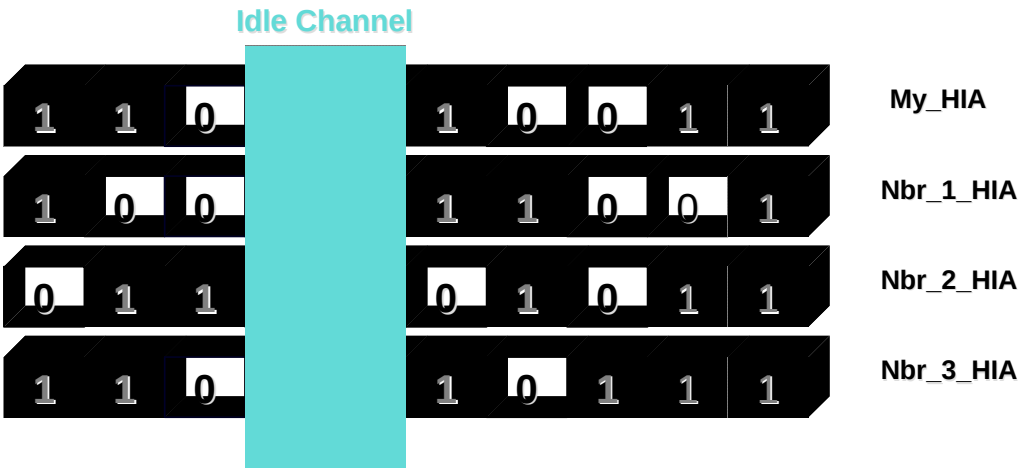


Figure 18: Idle Channel Selection from Hole Information Arrays

The target scenario consists of a set of primaries emit radio signals in a band, which is segmented into channels (we call them *frequency slots*), and a set of secondaries seek to self-organize into an ad hoc network using "holes" in the spectrum assigned to the primaries. As mentioned above, the problem has many variants and is very hard in general. Therefore, we shall make assumptions and design choices to simplify the problem. We assume that the primaries are chatty[6], that XG secondaries have two transceivers, one of which is always tuned to a dedicated *coordination channel*, and that they have a mechanism that can tell a secondary node from a primary one. Finally, we use a contention-based protocol for medium access rather than a scheduled scheme (e.g. TDMA) for reasons of simplicity.

An XG secondary node senses the channel and maintains a time-varying *Hole Information Array (HIA).* The HIA contains, for each channel in the band, an entry 0 or 1 indicating, respectively, whether the channel is free (a "hole") or is occupied (a "wall").

Each node periodically broadcasts its HIA as part of a control message called the *Hole Information Protocol (HIP)* packet. One purpose of the HIP packet is to serve as "beacons" or "Hello" messages for the purpose of discovering neighbors, as in many ad hoc routing protocols (e.g. OLSR). Specifically, a node considers another node as its neighbor if the moving window average of the number of received HIP packets exceeds a threshold. The HIP packets are sent on the dedicated *coordination channel* assumed allotted to the secondary nodes.

The HIP packet is also used for selecting the *idle channel.* Recall that this is the channel to which a node is tuned when it is not transmitting. Clearly, the idle channel for a node M must be chosen such that all 1-hop neighbors of M can transmit to M on it. That in turn implies that if *c* is an idle channel, then M and all of its 1-hop neighbors should have a hole in *c.* Thus, each node collects the HIA's from its neighbors, and finds overlapping holes, and assigns the idle channel for itself. This is illustrated in Figure 18. As shown there, the idle channel need not be a single frequency slot. It can be a contiguous set of slots.

Once the idle channel is selected, it is included in the HIP packet so that all its neighbors know. Also included is the transmission power of the HIP packet so that neighbors can estimate the path loss (based on received power) and the waveform used to listen to packet preambles. The selection procedure is specified in more detail in [XEP].

[6] We assume primary nodes are not silent for extended periods, and therefore, XG nodes can discover their presence.

We now describe briefly our MAC protocol. It is based on the well-known idea of Carrier Sense Multiple Access (CSMA), but the basic algorithm needs considerable modification in the presence of spectrum agility. A brief description is as follows. When a node S wants to send a data packet to node R, S first picks a set of frequency slots *F* that it perceives as holes (based on R's HIP packet) for both R and S. It then tunes to R's idle channel, say *I(R)*, and performs carrier sensing. If *I(R)* is free for more than a certain amount of time, S sends a Ready-To-Send (RTS) on *I(R)* and tunes to *F* and waits. R ensures that *F* does indeed reflect its holes and responds (if appropriate) with a Clear-To-Send (CTS) on *F*. R remains tuned to *F* for the DATA packet. Once S gets the CTS, it responds with the DATA packet and R responds with the ACK.

Unlike IEEE 802.11 MAC, the RTS/CTS is not effective for collision avoidance since each node may listen on a different idle channel. A node does not maintain a Node Allocation Vector (NAV) table based on the RTS/CTS packets overheard because it is not likely to hear such packet unless it is the intended recipient (RTS/CTS packets to nearby nodes will be transmitted over different frequency slots). Despite this, the RTS/CTS exchange is important for the selection of frequency slots and traffic flow coordination. However, better methods, perhaps based on TDMA are required going forward.

An issue that arises for large dense networks when the coordination channel is of low capacity is that the HIP packets may saturate it. To mitigate the problem, we have designed a lightweight protocol called *Rendezvous MAC (R-MAC)*. In R-MAC, a sender makes a prediction of the next time it will send a HIP-packet and includes this information as part of the HIP packet. Neighbors avoid transmitting on the "reserved time". No carrier sensing, RTS/CTS or complex backoff procedures are needed. The simplicity is possible due to the largely periodic, and hence predictable nature of the HIP packets.

While the above solution sketch for spectrum agility does enable opportunistic spectrum access, it is clearly leaves something to be desired. The need for an a priori dedicated coordination may not be realistic in many environments, and the question is if we can design a protocol that does not require it. It is also clear that CSMA has fundamental limitations with spectrum agile protocols and we need to investigate TDMA based algorithms, which is quite challenging. And if TDMA is used, there would be a need to disseminate hole information further than just 1 hop and this brings into question scalability issues.

5.3 Simulation Results

We used OPNET to simulate our XG system model and protocols running underneath a `legacy' MAC system. The results are presented here.

We used a CSMA MAC to test that our system – while being simple – provides a complete solution that is able to work with any existing and future MAC implementing the opportunity API specified in [XEP]. For example, a legacy MAC that is sensitive to timing information tied to the packet's transmission rate will need to specify their desired transmission rate to XG. If this rate is achievable, XG will create IDLE channels with that rate.

For simplicity, we implemented an unreliable MAC (no ACK) when a node transmits a packet if the channel is free or backs off for a random time if the channel is busy. Nodes also backoff upon completing a packet transmission to prevent some nodes from capturing the channel. Even with this very simple MAC we were able to show a good performance enabled by using XG.

Table 3 summarizes the default simulation parameters for both primary and opportunistic nodes. Unless stated otherwise, our simulations consist of 60 opportunistic and 10 primary nodes placed randomly in a square area of 0.6 miles by 0.6 miles. The primary nodes are fixed and not always active. When they are active, they continuously transmit packets using the ON/OFF cycle shown, and record the period of time that they experienced interference by opportunistic nodes beyond the maximum tolerance. Their

transmission range is equal to 2 km, which is consistent, for example, with current cellular systems. A random mobility model is supported for the XG nodes; their speed can vary from 0 - 10 mph, with nodes bouncing back each time they reach the area boundary. The results we present in this section, however, are limited to static XG nodes. Their transmission range is 250m, which is consistent with current WLANs such as those based on 802.11. Opportunistic nodes generated a new packet to transmit as soon as the previous transmission is completed. This corresponds to a saturation case, particularly challenging for an unregulated CSMA MAC without feedback-based load balancing. Each packet's destination is chosen randomly among the one-hop neighbors. We collect statistics on the total number of packets successfully received by each node.

Thus, we collect two main figures of merit: the total aggregated one-hop throughput (sum of all bits received by each node), and the maximum interfered time (the time a primary node measured interference above its tolerance).

Table 3: Default simulation parameters

General Parameters		**Primary Node Parameters**		**Opportunistic Node Parameters**	
Name	**Value**	**Name**	**Value**	**Name**	**Value**
Simulation area	0.6 x 0.6 miles	Bandwidth	19MHz	Speed	0-10 mph
Simulation time	200 s	Transmit PSD	-26 dBm/Hz	Coordination Channel	200 kHz
Base Frequency	2.3 GHz	Avg. On time	160 ms	Max Transmit PSD	23 dBm
Total Bandwidth	100 MHz	Avg. OFF time	180 ms	Target PSD (policy)	-62dBm/Hz
Unassigned Bandwidth	5 MHz	Transmit range	2km	Transmit PSD	0.625 mW/MHz
Background noise	-174 dBm/Hz			Transmit range	250 m
				Target SNR	12 dB
				Sensing period	8 ms
				Sensing history	256 ms
				Sense threshold	-138 dBm/Hz

Finally, the physical layer assumes some capabilities such as sub-noise detection (enabled by a DSP intensive synchronization stage). Thus, if a large processing gain is used, the sensing range could be smaller than the transmission range. The propagation model assumes an attenuation model proportional to the fourth-power of the distance.

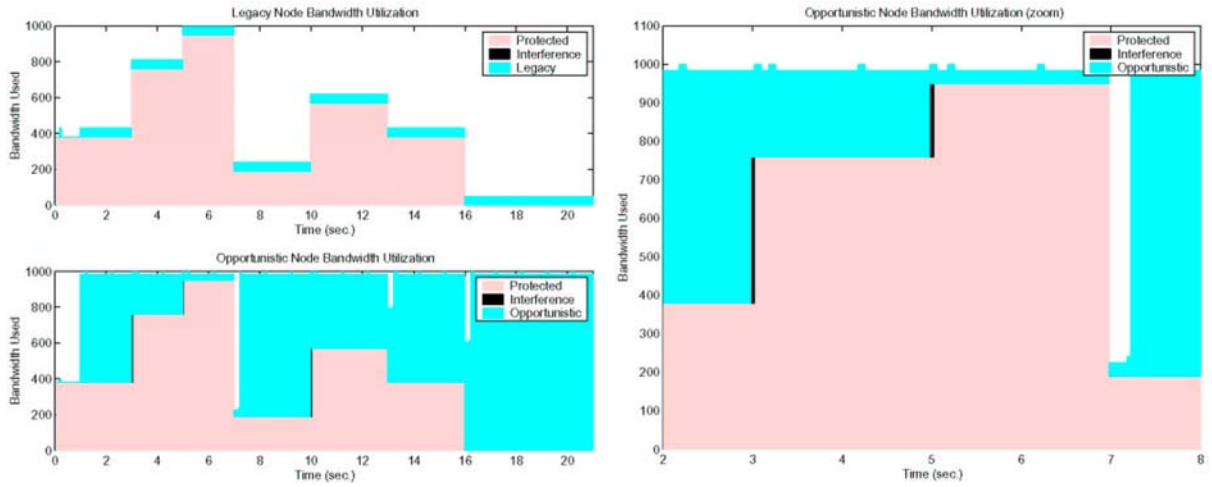


Figure 19: Spectrum utilization by Legacy (upper left) and XG (lower left) systems given the primary system utilization is 40%. A more detailed snapshot of the spectrum utilization by the XG system is shown on the right. Bandwidth is measured in terms of number of 100kHz channels used.

To show how our system exploits unused spectrum, experiments were conducted on a very simple network consisting of 4 nodes in a line, separated by 250m each. The primary nodes' tolerance was

decreased so that all the opportunistic nodes were inside the area where the primaries are protected by policy. Since we wanted to accentuate the negative impact of the coordination channel and modulation-type requirement, we increased the size of the coordination channel to 1.5MHz and the maximum transmit power of the opportunistic nodes to 40dBm.

Figure 19 shows how our algorithms take advantage of the unused portion of the spectrum. For these plots, we have divided the time into 40ms intervals and consider a frequency band used if a transmission occurred over any part of the 40 ms interval.

The upper left plot shows the way a legacy (i.e. current, fixed spectrum allocation) system – running the same MAC – used the frequency. The primary nodes are assigned 95 MHz of spectrum, so the legacy radios can use only the remaining 5 MHz. The frequency band has been divided in frequency slots of 100 KHz each, and the number of frequency slots occupied by each – and not the actual frequencies – is shown. The lower area represents the primary nodes utilization. For example, in the interval <0, 3> two primary nodes are active occupying 380 slots (i.e. 38 MHz). Similarly, in the interval <16, 21> no primary node is active. Averaging over the interval <1, 21> (i.e. after initialization) and over their assigned frequency bands, the primary nodes' present an utilization of 40%. Looking at the legacy node's bandwidth utilization, we can see the legacy nodes always use the same amount of bandwidth (the unassigned 5 MHz) and are constantly transmitting, with the exception of the interval <0, 1> (initialization) when they are only active for short bursts sending neighbor discovery beacons.

The lower left plot illustrates the way XG radios exploit the available spectrum. After the one-second-initialization period is completed, we see that XG fills the spectrum gaps almost completely, except for minute periods of time. Dark lines represent the periods where XG radios interfere with primary nodes. This interference is due to the latency on sensing the channel. It may take up to two sensing intervals (8 ms each) for the radio to detect that a primary node has become active, resulting on interference periods of up to 16 ms per each time a primary node becomes active. However, since our time step is 40 ms, we paint the whole 40 ms interval as `interfered'.

The right plot is a `zoom' of the lower left plot (XG spectrum utilization) for the interval <2, 8>. This allows a more detailed examination of the XG system’s behavior. First, we may notice that our XG system does not occupy the entire 1000 frequency slots all the time. Typically there will be a small gap, corresponding to the coordination channel's 15 slots. Only when HIP packets are sent (periodically every second and also event-driven upon primary nodes' activation at times 3s and 5s) XG will occupy the entire spectrum. Thus, the coordination channel and XG control packets reduces the DATA throughput achieved. Second, we see that upon a primary node's activation (for example at time 3 sec.), XG transmissions will interfere with primary nodes until their local sensors detect the primary node signal, and before any control packet is sent they will stop transmitting on the primary node's frequency band. This implies that they will not be able to communicate with any neighbor whose IDLE channel contains parts of that band (all the 4 nodes in our example). The XG nodes will then recompute their IDLE channel and include it together with their new HIA information in a HIP packet that is broadcasted after some random time (to avoid collisions from all opportunistic nodes sensing the primary node's signal at the same time).

Thus, in Figure 19 (right) we notice that the coordination channel is used after the interference period ends. So, from the time the primary node is detected until a HIP packet with new IDLE channel information is received, the nodes will not be able to reach neighbors (due to invalid IDLE channels).

The fact that we don't see white gaps in the spectrum occupancy at times 3 sec. and 5 sec. indicates that the nodes recover new information and resume transmission in less than 40 ms Thus, the periods of communication disruption in XG are small. Third, when at time 7 sec. primary nodes become inactive, we see that there is a period (around 300 ms) when the system is unable to fully occupy the spectrum.

Locally detecting this opportunity is not enough to use it - we need the neighbor nodes to be able to transmit over it. Therefore, XG nodes must wait until receiving neighboring nodes' HIA updates (in this case once every second) declaring the opportunity before recomputing their IDLE channel. In the figure, around time 7.3 sec. we see the HIP packets being sent, and only after that the IDLE channels are recomputed, another round of HIP packets with the new IDLE channel information are sent, and the XG nodes start using the new opportunities. Note that during this transition there is no transmission interruption since XG nodes tune to the new IDLE channel only after the HIP packet transmission has been completed. In between (i.e. after waiting for a spreading time to avoid collision, for the coordination channel to become free, and for the transmission delay) the XG nodes are still tuned to their last-advertised IDLE channel. Overall, XG reacts fast to new opportunities. It could even react faster if HIP packets are sent immediately after detecting new opportunities, similar to the case of losing opportunities being used by someone's IDLE channel, but this is not advisable since it may lead to congestion on the coordination channel. Thus, only critical (i.e. causing communication disruption) events trigger extra HIP transmissions.

The two lower curves of Figure 20 plot the total throughput achieved by the legacy and XG systems for different levels of bandwidth utilization by the primary nodes. For example, the experiment shown in Figure 19 corresponds to the 40% utilization data points. Since we set a large coordination channel of 1.5 MHz, it is expected that when the primary nodes utilization approaches 100%, the legacy system (able to transmit over 5 MHz) outperforms the XG network, which can only use 3.5 MHz for data transmission. It can be seen that under a bandwidth utilization of 40%, XG gives an order of magnitude performance improvement over legacy systems, even when assuming a generous allocation bandwidth allocation for the latter (5 MHZ of exclusive spectrum usage).

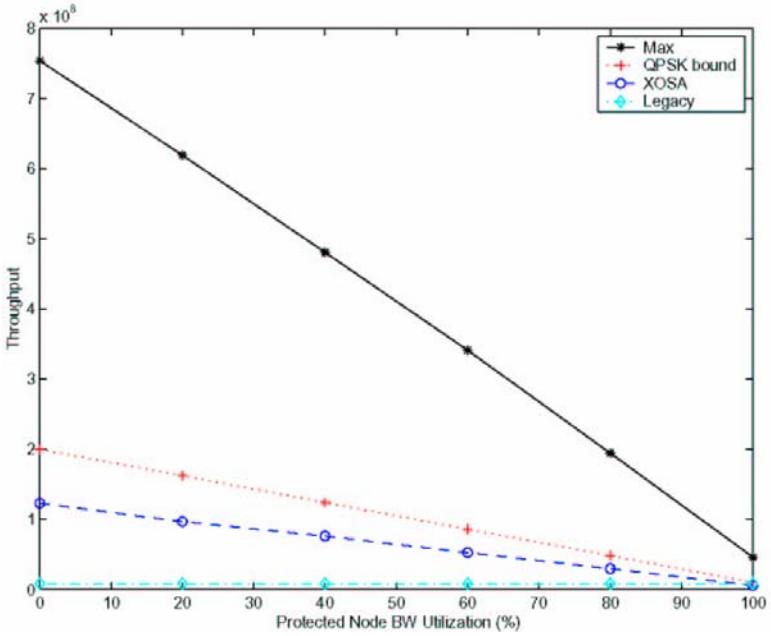


Figure 20: Comparing throughput achieved by XG and Legacy systems. The maximum achievable using a theoretical bound (based on Shannon capacity), and the maximum achievable if only QPSK modulation is allowed are also shown.

For comparison we are also plotting the maximum achievable throughput (i.e. capacity) computed using a theoretical model we developed, and described in an accompanying draft paper. We can see that our XG system is far from achieving the optimal value. In general, there are several factors preventing XG from achieving the maximum. Among them includes fairness (i.e. the optimal solution in the theoretical model gives exclusive access to the shortest links, which may not match the actual traffic demand),

using energy-inefficient modulation, not using long and highly efficient but computationally expensive codes. In this example, the main reason for the difference was our use of energy-efficient modulation.

We assumed bandwidth efficiency of 1 bps/Hz or less, therefore saving energy and extending the network lifetime, an important factor in mobile networks. To corroborate this, we are also plotting the maximum throughput achieved if the waveforms used are restricted to QPSK modulation. As we see, our system is within reason of the maximum achievable by an energy-efficient system for this simple topology. The impact of more complicated topologies on this MAC is the subject of future work.

The above results are representative of the types of studies that can be performed using the XEP. Using the XEP, we have investigated the relative merits of underlay vs. overlay, and the impact of topology control and the interference tolerance of the primaries on the overall network throughput. Preliminary results from these studies are available in the form of a draft report included with the deliverables.

6 Technical Interactions

In support of the DARPA XG program goals of transitioning the technology developed to GOTS and COTS systems, BBN had numerous interactions with the XG working group, industry, academia, and regulatory bodies. We present a brief summary of significant interactions and related activities below.

6.1 Working group meetings and other briefings to XG performers

BBN has engaged the XG Working Group participants in the development and refinement of the XG RFCs. We participated in and provided briefings in two working group meetings, one of which was hosted by BBN in our Cambridge, MA facility. The working group process was an integral part of the development and release of the XG RFCs; it provided the primary mechanism through which to capture the concerns of the stakeholders (XG Government and SI team participants).

A notable benefit of the WG process was the generation of policy use cases, which provided concrete contexts for the development of the policy language as well as useful test cases for the policy conformance reasoner software.

In addition we have had several interactions via phone and email with all three SI teams, as well as with Alion Science, in regard to the policy conformance software. In addition, we gave several briefings to a team from Shared Spectrum Corporation that visited BBN for a daylong discussion on implementation and integration of the policy technology.

6.2 Website and mailing lists

BBN maintains a website to support the XG working group activities as well as to disseminate to the public, with Government approval, key results of the XMAC and XAP contracts. The policy conformance reasoner software has been made publicly available through this website.

In addition, BBN maintains three mailing lists for XG in order to promote discussions and in order to make announcements related to software and RFC releases:

- **xg:** limited to XG Working Group members
- **xg-panel:** limited to XG Industry Panel members
- **xg-public:** open to the public

6.3 External briefings and interactions

FCC: BBN provided a briefing to the FCC in support of DARPA's XG policy technology briefing to the FCC. We received a favorable response, and the audience expressed interest in seeing demonstrations of XG policy technology.

NSF Programmable Wireless Networks: BBN has briefed the academic research community at the NSF Programmable Wireless Networks workshop.

XG Industry Panel: BBN participated in the discussions of the XG Industry Panel in which DARPA briefed the industrial research community on XG policy technology.

Philips Research: We have a technical interchange with a group at Philips Research that is actively doing research in the area of spectrum sharing etiquette, and spectrum measurements in the context of IEEE 802.11 standardization activities.

6.4 Academic collaborations

Northeastern University: We have been working with Prof. Karl Lieberherr and Prof. Ravi Sundaram of Northeastern University in the area of investigating the application of Aspect Oriented Programming techniques in XG systems. Another area of discussion involved the modeling of secondary spectrum markets using truth-telling auction mechanisms. Pengcheng Wu, a student of Prof. Lieberherr, participated in the XAP effort as a summer intern at BBN. Overall, this interaction has positively influenced our development of the XG Abstract Behaviors.

Virginia Tech: We have collaborated with Prof. Amitabh Mishra of Virginia Tech to develop a proposal to the NSF to apply XG policy technology in the context of reuse of 3G cellular spectrum.

Lehigh University: Recently, towards the end of XAP contract, we have had brief email interactions with Prof. Donald Hillman of Lehigh University. The area of discussions included the further development of the XG ontologies (the structure of authority in particular) and the formulation of the dynamic spectrum assignment problem as a deductive database problem. The ideas from these discussions are worthy of further exploration in Phase 3.

7 Summary and Future Work

The current practice of statically allocating spectrum is widely recognized as inefficient. Although virtually the entire spectrum has been allocated to services and much of it is assigned to users, measurements within the XG program and elsewhere have demonstrated that large portions of spectrum are not being used at any given time or location. Regulatory agencies are now considering opening up spectrum for opportunistic access.

Furthermore, while opportunistic access is of tremendous value in the civil sector, it is vital for the military. Since the military is increasingly in need of (and reliant upon) high bandwidth communications, maximizing use of spectrum is an essential military capability.

The primary objective of XG is to increase spectrum utilization by enabling devices to opportunistically utilize spectrum whenever and wherever it is available. A secondary effect of this capability will be to greatly reduce spectrum-planning time prior to deployment. The XG program is developing technologies that fall in two classes:

1. ***Spectrum Agility:*** Actual spectrum usage varies with location and time. Devices must, therefore, possess *spectrum agility*:
    a. They must be able to acquire awareness of the spectral environment
    b. They must be adaptive in the use of spectrum
    c. They must be able to coordinate the use of spectrum with other devices.

    The XG program has developed new technologies and systems that leverage recent advances such as wideband sensing and software radio technology to provide spectrum agility to wireless devices.
2. ***Policy Agility:*** The use of available spectrum is subject to policy rules that vary with frequency, type of service, type of operator, location, and the regulatory jurisdiction. Furthermore, it is a challenge to manage the accreditation of agile devices for all the uses they are capable of across regulatory regimes. Thus devices that make use of opportunistic spectrum access must possess *policy agility*: They must be able to adapt their operation to opportunities that are authorized by policy, using verifiable approaches. The XG program has, therefore, developed a trusted *policy language framework* to provide policy agility to wireless devices.

7.1 BBN's Accomplishments in the XG Program

BBN Technologies is performing the role of architect in the XG program. BBN has developed a series of XG Requests for Comments (RFCs) that document the vision, architecture, as well as technical specifications for XG systems. Following are a few of our accomplishments during XG Phases 1 & 2:

- Produced four RFCs that define aspects of the XG architecture
- Developed a policy language framework that will eventually enable XG-capable radios to adhere to relevant policies as they dynamically choose where and how to use available spectrum. An additional benefit of this machine-readable policy language is the ability to define "temporary" policies to which radios would conform – thus encouraging experimental use of spectrum.
- Developed reference Policy Conformance Reasoner (PCR) software that demonstrates how a radio system can check whether its intended emissions conform to policy (as specified in the defined policy language). The PCR can also search the policy constraint space for opportunities that are allowed by policy. A version of this software has been delivered to all XG performers

and at least one system integration team has integrated the PCR with their system and demonstrated the combined capabilities in a limited form.

- Produced a simulation environment that allows us to estimate the possible performance gains of various XG spectrum sharing algorithms and protocols.

In addition BBN has supported DARPA's goals of dissemination of key aspects of the XG architecture to the public through briefings at the FCC, NSF Programmable Wireless Networks community, and the XG Industry panel. BBN has also briefed, DARPA personnel from other offices (IPTO), and researchers from the industry (Philips Research), and academia (Northeastern University and Virginia Tech). Furthermore, BBN has worked with the XG system integration teams to support transition of the XG policy technology, by providing access to BBN-developed software as well as through briefings. BBN has maintained a website and three mailing lists for this purpose.

In Phase 2 BBN developed an architectural framework for XG systems, and investigated the core research issues for policy-agile XG operation. We demonstrated proofs-of-concept of the technologies we developed through a prototype policy conformance reasoner software, notional XG policy examples, and basic ontologies populated with concepts that are sufficient for a small range of interesting XG policies. Effectively transitioning the XG policy technology into DoD systems requires further development effort that can directly build upon our results and deliverables from Phase 2.

Milestone	**Pre XG**	**End of Phase 2**	**Early Phase 3**	**End of Phase 3**	**Post XG**
Time		**Dec 04**	**Aug 05**	**Feb 07**	**Oct 08**
Objective Implementation		Demo XG Policy Engine	SI Integration	SI Demo	DoD Transition (JTRS)
Spectrum Policy Representation	English Text	Ontology (XG Notional) 100 concepts 50 policies 200 facts	Ontology (XG Domain) 300 concepts 150 policies 1000 facts	Ontology (XG Domain) 1000 concepts 500 policies 5000 facts	Ontology (DoD Domain) 5000? concepts 2000? policies 20000? Facts
Opportunity Query	Spectrum: None IETF: Point spec, 7 Dimensions	Fully bound spec (a region) 13 Dimensions	Partially bound spec, 1 unbound variable	Partially bound spec, n unbound variables	Utility Spec. 100? Dimensions
Policy Encoding	Text Procedural language Hardware	OWL CLIPS String/socket	OWL CLIPS CORBA	OWL Custom Reasoner CORBA	Web Services Custom Reasoner CORBA Hardware assist
Policy Loading	Built-in	Off-line merge On-line load from local/network store	+ Dynamic Merge	+ Performance enhancements	+ Secure load
Real-time Reasoning	None	Validation (120ms)	Validation (10ms) Search (5s)	Validation (10ms) Search (1s)	Optimization (10s) Search (100ms) Validation (1ms)
Accreditation	Case by Case Policy sets X platforms	Architecture reduces the number of accreditations		Accreditation roadmap	Policy sets + Platforms + Reasoner

Figure 21: XG Policy Technology Roadmap

As the XG program enters system integration, test, and demonstration in Phase 3, we have developed a roadmap for XG policy technology, in consultation with the DARPA PM for XG. The roadmap, illustrated in Figure 21, summarizes the current status of the technology, and traces the development needed in Phase 3 and beyond. The key milestones on the roadmap for this technology in Phase 3 are integration and demo with XG systems. The integration and eventual transition to DoD systems must be accompanied by further development of the language framework, in particular, through population of

the policy ontologies with a larger repertoire of relevant concepts, and the creation of tools suitable for policy administrators and defense spectrum planners.

We have also converged on a proposed architecture for XG system integration during Phase 3 illustrated in Figure 22. This architecture clearly identifies the relationship of the policy engine (shown in green) within the XG system – we believe that BBN's XG policy conformance reasoner technology fits cleanly into this architecture. Also note that the clean architectural separation of policy from the system enables changes and innovations in either side to proceed independently.

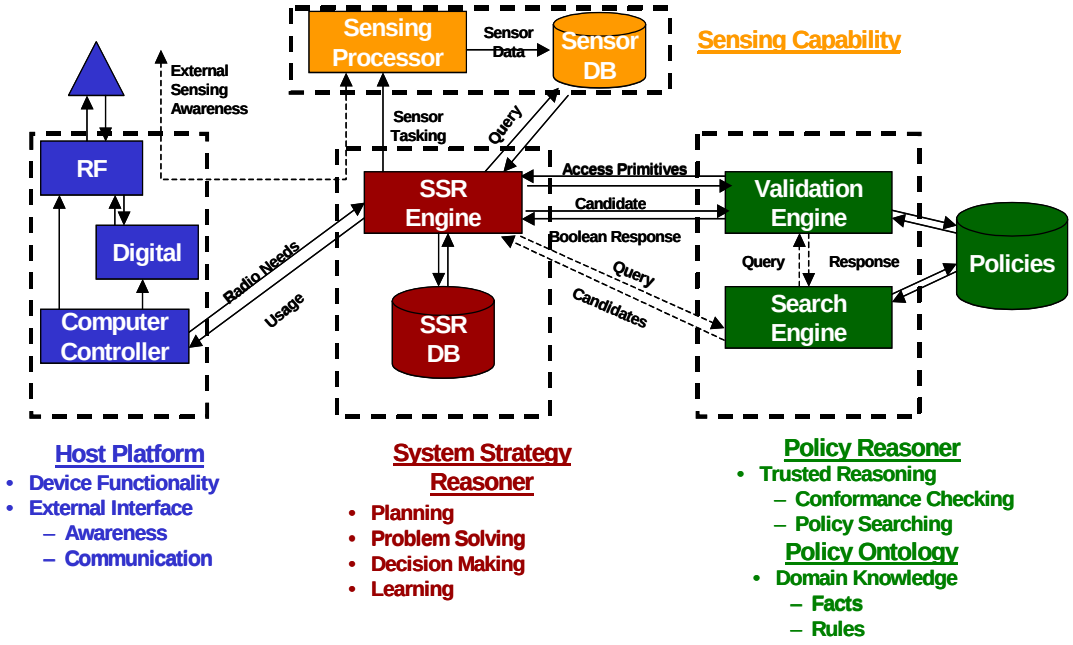


Figure 22: XG System Integration Architecture

This architecture suggests that it would be desirable to develop additional primitives to access the sensor database and integrate them with the policy conformance reasoner.

7.2 Future Work

Follow-on work that builds on our existing work will be required in order to support the XG program in Phase 3 and beyond. We believe the following tasks are critical during Phase 3 of the XG program:

- **Policy Conformance Reasoner (PCR) Software Enhancements:** The current policy conformance reasoner (PCR) software was developed as a set of functional prototypes. It includes an initial release of the Policy Validation Engine (PVE) software and an early proof of concept for the Policy Search Engine (PSE). Both of these systems require further development to add robustness, provide programmatic interfaces, and demonstrate substantial performance enhancements as we make them ready for integration into a target XG system. Furthermore, they must be tailored to the requirements of the particular XG system platform of the down-selected Phase 3 system integration team.
- **Policy Language and Content Representation Framework Enhancements and Authoring Support Tools:** As the XG policy technology leaves the laboratories of the XG performers and is transitioned into DoD systems, it is critical that basic tools exist to enable domain users such as system administrators and spectrum managers to create, manage, check, and provision policies for XG systems**.** Templates for common policy

idioms can make it easier for policy creation and management. Furthermore, the ontologies must be populated with enough content, and reorganized for more processing efficiency if needed, so that a wide range of useful policies for XG systems can be readily expressed.

In addition, the following important tasks must be considered before XG technology would be ready for transition and deployment into DoD networks.

- **Certification Roadmap:** Before XG systems can be deployed in the field - both within the US and in other countries - they must be certified for use by the relevant authorities. A study, possibly modeled on prior work on the certification of software defined radios, is required to identify the issues involved in certifying XG systems.
- **Embedded PCR Study:** A study is needed to analyze the feasibility and benefits of building a reasoning and inference engine in low-cost embeddable hardware instead of software. An embedded implementation may potentially offer performance, cost, or ubiquity benefits.
- **Security and Trust Model Evaluation:** A study is required to investigate security and trust threat models for XG policy. While XG radios are inherently as secure as software radios and their host OS and networking stacks, we need authentication to ensure that the rules for how the radio may operate in different conditions are indeed the policies issued by the responsible authorities.
- **Network Strategy Reasoner:** As policy-agile XG systems are deployed and integrated into the network-centric force, a study on how to extend XG methodologies to the realm of a dynamic network (beyond just spectrum) management will be beneficial.

References

1. [XGV] The XG Vision RFC
2. [XGAF] The XG Architectural Framework RFC
3. [XGAB] The XG Abstract Behaviors RFC
4. [XGPL] The XG Policy Language Framework RFC
5. [XEP] The XG Evaluation Platform software documentation
6. [PCR] The XG Policy Conformance Reasoner software documentation
7. [SPTF] Spectrum Policy Task Force Report, FCC ET Docket no. 02-135, November 2002, http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-228542A1.pdf

[DP] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object Oriented Software, ISBN: 0201633612, Addison-Wesley, 1995.

Appendix A
XG Vision
Request for Comments

Version 2.0

1 About this document

This document describes the vision for the DARPA neXt Generation (XG) communications program. It lays out the motivation for XG and its scope, presents the key concepts underlying XG, and describes an approach for defining XG.

This document is a Request For Comments (RFC). Accordingly, an important purpose of this document is to obtain feedback from the community at large, and refine the ideas here based on the feedback. In other words, the XG vision will evolve, and this document reflects a snapshot in that evolution.

A number of other RFCs related to XG exist, or are being planned. The complete XG family will include the following:

1. *XG Vision RFC*. This document.
2. *XG Architectural Framework (AF) RFC*. The AF RFC presents the architecture, system components, and a high level concept of operations for XG communications.
3. *XG Abstract Behaviors RFC*. The abstract behaviors RFC identifies key behaviors that must be implemented by an XG system, organizes them, and describes the behaviors.
4. *XG Policy Language RFC*. The XG policy language RFC describes the policy specification meta-language for implementing machine-understandable policies.

This document is the “highest level” RFC, and should be read before any of the others. Its focus is on *what* XG is, in terms of its capabilities and scope, rather than on *how* XG will work. In a sense, this may be considered an evolving “executive summary” of the DARPA XG program. In short, it is intended to be a one-stop-shop for answering “What is XG” type questions.

2 An Introduction to XG

The Defense Advance Research Projects Agency (DARPA) neXt Generation (XG) communications program is developing a new generation of spectrum access technology. This section presents the motivation behind XG and its overall goals.

2.1 Motivation

There are two significant problems confronting wireless communications with respect to spectrum use:

- *Scarcity*. The current method of allotting spectrum provides each new service with its own fixed block of spectrum. Since the amount of useable spectrum is finite, as more services are added, there will come a point at which spectrum is no longer available for allotment. We are nearing such a time, especially due to a recent dramatic increase in spectrum-based services and devices.
- *Deployment difficulty*. Currently, extensive, frequency by frequency, system by system coordination is required for each country in which these systems will be operated. As the number, size, and complexity of operations increase, the time for deployment is becoming unacceptably long.

Both problems are a consequence of the centralized, static nature of current spectrum allotment policy. This approach lacks the flexibility to aggressively exploit the possibilities for dynamic reuse of allocated spectrum over space and time, resulting in very poor utilization and *apparent* scarcity. It also mandates a priori assignment of spectrum to services before deployment, making deployment difficult.

Preliminary data indicates that large portions of allotted spectrum are unused (refer the Spectrum Policy Task Force report). This is true both spatially and temporally. That is, there are a number of instances of assigned spectrum that is used only in certain geographical areas, and a number of instances of assigned spectrum that is only used for brief periods of time. This wastage is bound to increase in future – spatially, due to the increasing localization of propagation due to radio devices moving up in frequency, and temporally due to the proliferation of services that are highly bursty in nature.

Studies have determined that even a straightforward reuse of such "wasted" spectrum can provide an order of magnitude improvement in available capacity. It can be concluded that the issue is not so much that spectrum is scarce, but that we do not have the technology to effectively manage access to it in a manner that would satisfy the concerns of current licensed spectrum users.

In order to address the scarcity and deployment difficulty problems, XG is pursuing an approach wherein static allotment of spectrum is complemented by the opportunistic use of unused spectrum on an instant-by-instant basis, in a manner that limits interference to primary[7] users. In other words, the basic idea is this: a device first "senses" the spectrum it wishes to use and characterizes the presence, if any, of primary users. Based on that information, and regulatory policies applicable to that spectrum, the device identifies spectrum opportunities (in frequency, time, or even code), and transmits in a manner that limits (according to policy) the level of interference perceived by primary users. We term this approach *opportunistic spectrum access.*

Opportunistic spectrum access also provides far easier deployment, or rapid entry, into regions where spectrum has not been assigned. Only minimal prior coordination is necessary, greatly easing the restrictions to meet the deconflicting requirements. This is helpful both in civilian applications such as the entry of a wireless LAN technology in less developed regions, and in military operations requiring high tempo and quick reaction time.

[7] Users that are licensed to use the spectrum in question, subject to regulatory constraints.

The fundamental change from legacy systems is that the management of spectrum is now placed in each radio, where it can assess the actual situation at each instant in time, rather than have to be deconflicted in advance for any possible situation of time, position, signal, propagation, etc. Only a few of these constraining conditions will be present at any one time, and these are the only ones that need be considered in developing interference avoidance tactics. The radio itself is best positioned to be aware of these conditions.

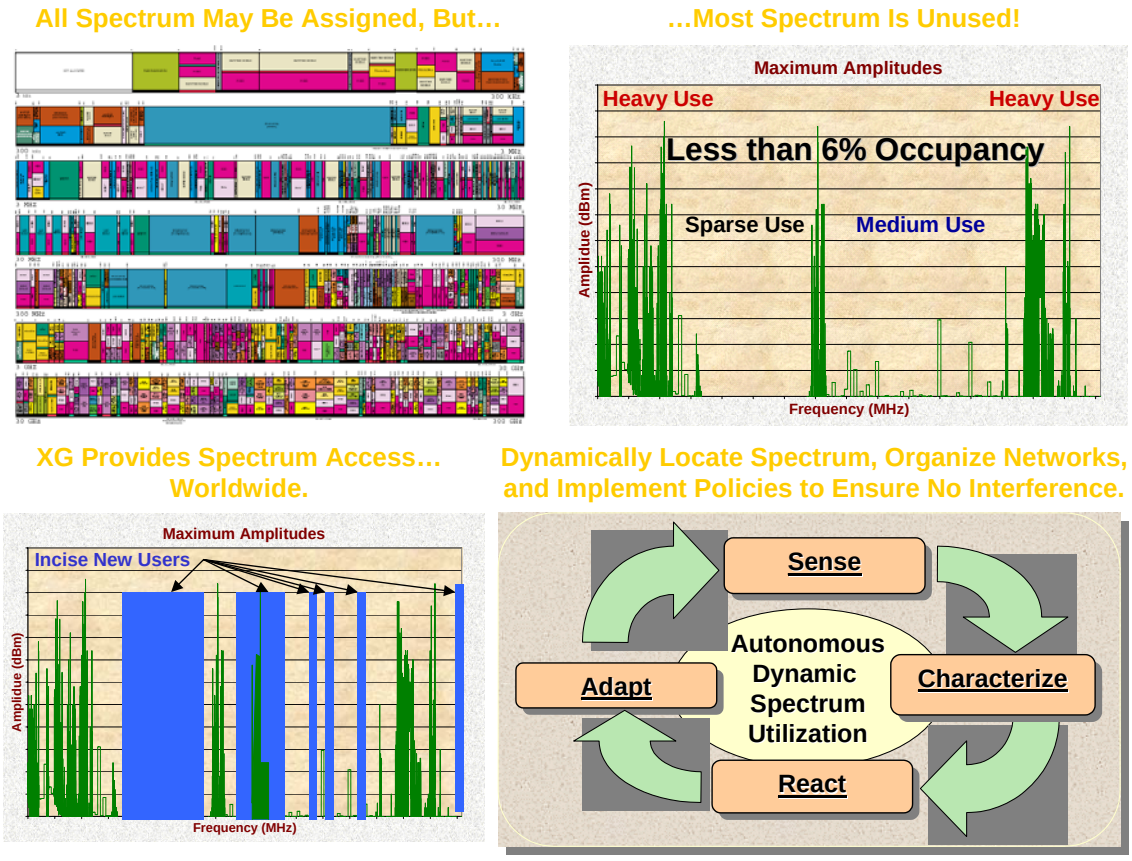


Figure 23: Spectrum is wasted. Opportunistic spectrum access can provide 10x improvement by reusing wasted spectrum.

While conceptually simple, the realization of opportunistic spectrum access is highly challenging. Several problems must be solved: sensing over a wide frequency band; identifying the presence of primaries and characterizing available opportunities; communication among devices to coordinate use of identified opportunities; and most importantly, definition and application of interference-limiting policies, and utilization of the opportunities while adhering to such policies.

Let us consider the last point further, as it is complex and significantly impacts the scope of our vision and approach. The ability to sense and transmit on unused spectrum, or *spectrum agility*, is doubtless the central capability required. However, the true potential of this new approach can be exploited only if in addition to spectrum agility, we provide *policy agility* – that is, a way by which the policies controlling the behavior can be dynamically changed. That is, policies are not embedded in the radio, but can be loaded "on-the-fly". Policy agility allows adaptation to policies changing over time and geography. Further, technology (spectrum agility) can be developed in advance of policies. This is important for breaking the chicken-and-egg dilemma that exists today, where regulatory bodies must wait for technology before drafting policies and technology must wait to see what the polices will look like.

The central role of regulatory policy in achieving opportunistic spectrum access suggests that a fresh look at the process by which policies are conceived, specified, and applied is needed. This is where XG is unique. The XG vision includes not only

the development of technology for opportunistic spectrum access, but also the development of the concepts, tools and standards for incorporating a totally new "software-based" policy regime that allows policies to be decoupled from the implementation and changed dynamically.

The use of policy agility using *machine readable* or *machine understandable* policies is depicted in figure 24. Starting from the left, spectrum policies are encoded in a machine interpretable form and loaded into the XG device. The XG device then operates in accordance with its interpretation of these policies.. Policies may be loaded using smart media or over the Internet. In order to change the policies we simply need to load a new version. For instance, operating in a different country would require merely downloading from a different web-site or new smart card.

In general, if a radio were not policy agile – even if it is a software-defined radio that is capable of doing opportunistic spectrum access – then each policy change would require re-design, re-implementation, and re-accreditation. And this potentially needs to be done for each configuration of a system (e.g JTRS UAV, JTRS vehicle mount, etc.). Further, accreditation is an *m x n* problem, that is, for each of *m* policies, and each of *n* radio types/configurations, a separate accreditation needs to be done. On the other hand, with policy agile radios regulators need to accredit once (accredit based on ability to correctly interpret machine-understandable policies), and can change policy dynamically as the situation changes.

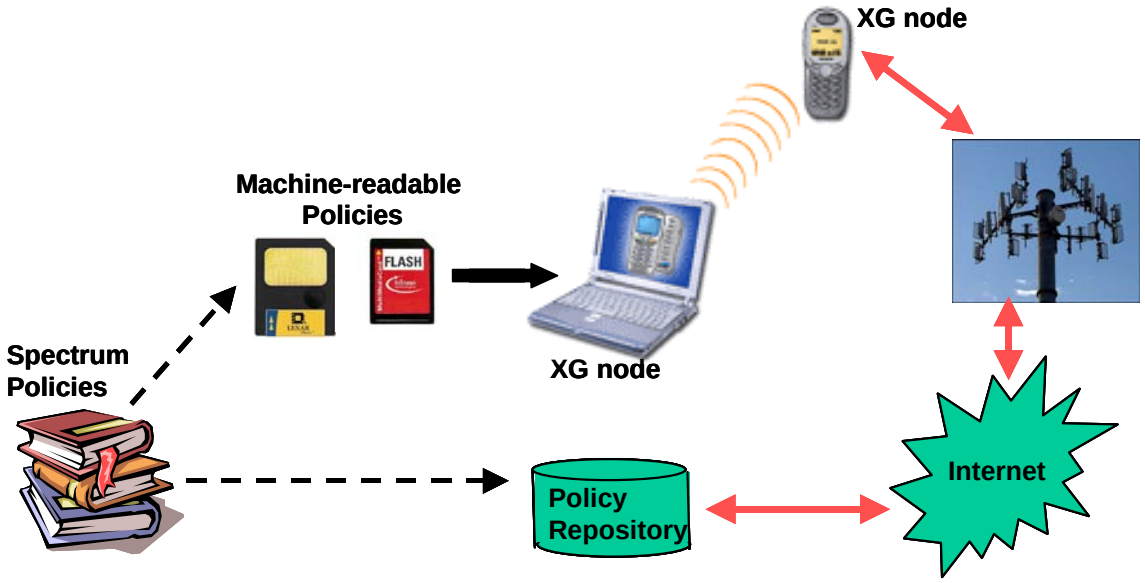


Figure 24: Machine understandable policies. With this, changing the policy merely means loading a different flash card or downloading anew.

Although recent years have seen some of the components for opportunistic spectrum access mature (e.g. software radios), we are a long way from a prototypical system. Further, no work exists in the area of decoupling the policies from the implementation. This yawning gap between current state-of-the-art and what is required for opportunistic spectrum access is the motivation behind XG. XG will develop the enabling concepts and technology for opportunistic spectrum access in an interference limiting manner. Further, it will develop an architectural framework for specifying and applying XG policies to XG devices. With these developments, XG anticipates revolutionary advances in network capacity and ease of entry while enabling a novel, highly flexible regulatory regime by providing the languages, idioms, tools, and concepts for the specification and application of machine-readable policies.

2.2 Goals

The goal of the XG program is to solve the problem of opportunistic spectrum access in its totality. At the highest level, there are two sets of goals.

1. *Develop the enabling technologies for opportunistic spectrum access*. This includes providing certain key behaviors such as sensing and characterizing the environment, identifying and distributing spectrum

opportunity information, and allocating and using these opportunities commensurate with the demand. Such solutions would typically be implemented as part of an XG radio device.

2. *Develop a long-lived framework for managing the key aspects of radio behavior through flexible application of policies*. In order that the radio be policy-agile, we require a framework in which policies are written in a way that can be interpreted by the radio, and the radio is able to exploit such expression of policies.

The remainder of this section elaborates on these goals, with particular emphasis on the second goal.

For each of the behaviors mentioned in item 1 above, it is likely that there is more than one solution. Therefore, one objective is to ensure that our framework is flexible enough to permit multiple solutions within a single abstraction of how XG should operate. In other words, the framework should allow diverse solutions to co-exist while sharing a core set of behaviors.

Another goal is to keep the core behaviors distinct from the innovations that may implement the mechanisms in different ways. This would be analogous to secure kernels – that is, inside the boundary, we can be sure of what is happening and can trust it whereas outside this boundary there is room for innovation. The challenge is to make it so that only the core set of behaviors "inside the boundary" is relevant for regulatory approval. This idea will be discussed in more detail in section 10.2.

Achieving policy agility requires the decoupling of policies from behaviors and their implementations. This implies the need for a standard way of expressing policies across the XG program, and in future, across the XG-compliant nodes. Thus, a goal is to develop a language that can provide a suitable mechanism for policy expression and interpretation.

A key goal of the XG vision is *traceability,* that is, the ability to associate each emission with a policy or a set of policies that permit this emission. Traceability would be a valuable feature that will help address the thorny verification and validation problem.

Information assurance, while important, is not a focus of the XG effort. The goal is to be consistent with existing information assurance architectures (such as red/black separation), and not invalidate existing constructs. However, denial of service issues, at least on the XG control channel, must be considered. Design of protocols must pay heed to at least the well-known denial of service threats.

Finally, a challenging goal is to ensure that XG is not unduly influenced by how we plan to implement the solution today, and is instead a flexible framework that can be used for decades after the XG project is completed at DARPA. In other words, we need an overarching technology that can be separated from, and be managed above the level of individual radio approvals. The longevity of the Internet Protocol is proof that such frameworks are possible.

In sum, our vision is to enable two new regimes:

- A new spectrum access behavioral regime consisting of technologies that sense, characterize, and utilize spectrum opportunities in an interference-limiting manner.
- A new regulatory control regime consisting of methods and technologies for controlling such opportunistic spectrum access behaviors in a highly flexible, traceable manner using machine understandable policies.

3 The XG Approach

A key facet of our approach is the *decoupling* of policies from behaviors and behaviors from protocols. To illustrate this, consider current practice, for instance IEEE 802.11. The 802.11 developer has the IEEE standard as the reference point for the physical and MAC layer implementation. The IEEE 802.11 standard describes the physical- and MAC-layer protocols. The *behavior* (the "what"), is implicit in, and tied to the *protocol* (the "how"). Similarly, the *policy* of sharing the unlicensed band, in particular the "good citizen rules" such as maintenance of power spectral density to within a certain value, is expressed as part of the power control and spreading mechanisms and is "embedded" within the implementation.

Our approach is to decouple these elements as shown in figure 25. By this, we mean that policies, behaviors and protocols are defined separately with some kind of "connections" defined between them. For instance, in the context of the above 802.11 example, using this approach, there would be three "specifications" – one defining the usage policy, one defining the behaviors, and one defining one or more protocols that implement each behavior. Each specification would be independently changeable within the bounds of its parent abstraction's specification.

While this might be overkill for 802.11 and other legacy "static spectrum" systems, it is of great value for

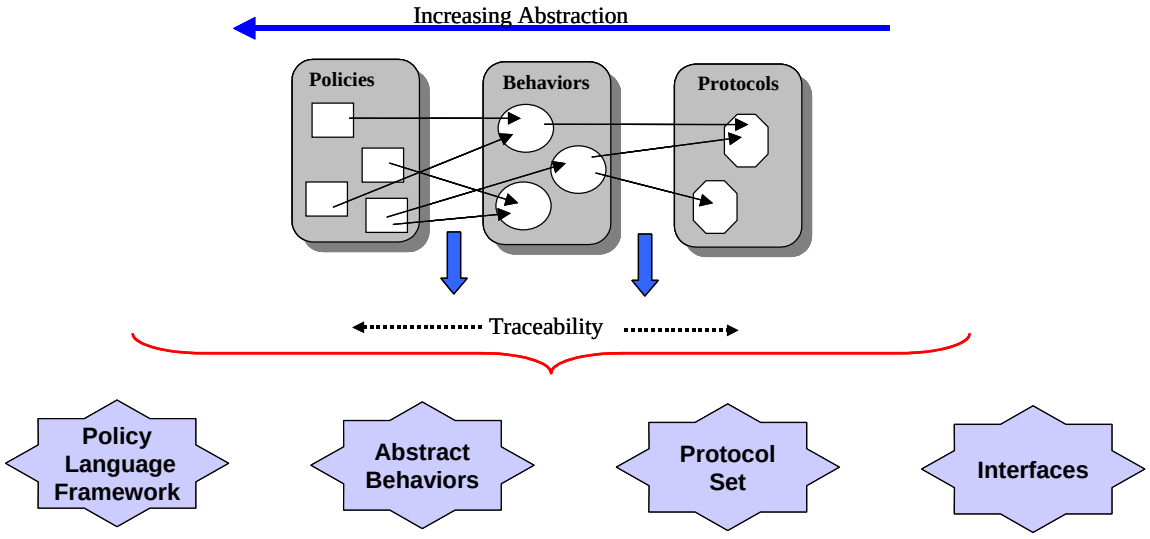


Figure 25: This illustrates the decoupling of policies, behaviors, and protocols, along with traceability and the four components of our framework.

opportunistic spectrum access. Decoupling allows adaptation to policies that vary over time and geography. Technology can be developed in advance of policies, and worldwide deployment would be greatly simplified. Furthermore, sub-policy management, as required in secondary markets, is easier. Policies no longer have to reflect the common denominator of competing technologies, and can be tailored to the diverse system capabilities expected for opportunistic spectrum access. Decoupling behaviors from protocols allows us to control *what* needs to be done separately from *how* it is implemented, resulting in a cleaner and more flexible architecture. Indeed, we argue that a decoupling approach is not just beneficial but pretty much a requirement for harnessing the full potential of opportunistic spectrum access.

Thus, a key aspect of the XG approach is that it is *policy controlled*. Policies are the only things that XG users, regulatory bodies, and foreign governments need to concern themselves with in order to control and predict the behavior of an XG system. As mentioned earlier, when an XG system is to be deployed in, say, a foreign country, we only need to supply a policy script (say in a memory card) as input to the XG system and the XG system behaves accordingly. This is similar in operational detail to the loading of a new configuration file, except that the policies are far more general and complex than the assignment of parameters to variables and the policy meta language needs far more expressive power than a typical configuration file.

We see policy as *constraining* (but not *specifying*) either the implementation details of the protocol or radio, or their performance. For example, a policy constraint could be that one must vacate an occupied frequency in *t* seconds. This policy is not dictating the performance of the protocol, or the design of the MAC, but it is constraining the solution. If for example, it takes an XG radio more than *t* seconds to acquire the signal, or to sample the band, the XG radio would not be able to abide by the stated policy, and would therefore have to avoid the band that was so constrained. In many ways, policies specify what "not to do". The XG radio and protocol implementations are free to operate in any way they want as long as they abide by policy.

A central notion in this approach, and a notion that is enabled by this approach, is that of *traceability*. Behaviors, preferably one or more of a core set of abstract behaviors, should be traceable to policies. This provides two advantages: first, it helps make the verification of new policies easier, and second, when an XG radio is deployed

in a new region, it is easier to affirm that the XG system will behave in a certain way. It allows us to accredit based on ability to correctly interpret and implement policies. In other words, traceability ensures that abstract behaviors (implemented by a specific XG system) can be validated against the policy – this is key to accreditation. Lack of traceability is a weakness of today's software-defined radios. For instance, JTRS radios on UAV, handheld, and vehicle-mounted systems have to accredit independently. By being policy controlled, XG systems will allow a single accreditation to cover all cases.

The XG approach has a number of advantages in comparison to the traditional way of architecting systems (such as SUO). First, the user of XG-enabled radios has far more and far easier control of the system behavior which can be quickly adapted to the diverse environments that are likely to be encountered. Second, as mentioned earlier, it provides for traceability to help the accreditation process. Third, one can use systems based on different set of assumptions simply by incorporating these assumptions in the policy (e.g., an XG system that relies on chatty primaries for sensing can be used with a policy that allows its use only in bands with known chatty nodes). This allows incremental development by progressive relaxation of assumptions. Finally, any misgivings about the use of XG with respect to existing services within a particular band (for example when there is zero tolerance for interference) can be mitigated by simply having a policy that forbids the use of non-primary emitters in that band, rather than redesign the hardware and software to do that.

We note that an alternative approach for XG would be to develop waveforms that optimized the performance of a radio, and to embed the control protocols within the radio as part of its design. This is similar to the approach adopted for several other DARPA communications programs, such as the Small Unit Operations (SUO) radio and Future Combat System Communications (FCS-C). While this approach would demonstrate the feasibility of the XG technology, it would not provide a *framework* that would enable the implementation of these features within a broad range of radios, nor would it establish a basis from which broad regulatory approval could be obtained. The process would still be one radio at a time, and would not advance our objective of an overarching technology that could be managed above the level of individual radio approvals.

Our approach requires the definition of four key components, as shown in the bottom of figure 26: policy language framework, abstract behaviors, protocols, and interfaces. National and international regulatory environment is a long-term process, and one that can not instantly adapt to changing concepts and approaches. Therefore we want to approach these communities with a simplified and generic set of *abstract behaviors*. We also need to characterize the control over XG policies – thresholds and rules of operation – by a *policy language*. Policies are expressed in a machine-understandable language. Abstract behaviors are instantiated by *protocols*, the specification of which will allow for interoperability. Fourth, in order that the framework be agnostic to the "best" solution for each function, and to allow for progressive, independent refinement of each function, *interfaces* (APIs) are a key element of the framework.

In the remainder of this section, we shall look more closely at our vision for the policy language framework, and abstract behaviors. We note however, that this is only an overview, and that further detail will be presented in the other RFCs in the XG RFC family.

3.1 Policy Language Framework

The policy language framework has four objectives: developing a language structure that is rich enough to adequately express XG use cases, allow for machine "understandability", support inference and reasoning capabilities, and be flexible/extensible enough to be long-lived. We note that it is not the intent of our framework/language to be able to capture *all* spectrum policies, only *XG related* policies.
To set the context for the policy language, we first consider the "big picture" – that is, the actors and roles involved in policy based control of radios. Our vision is depicted in Figure 26.

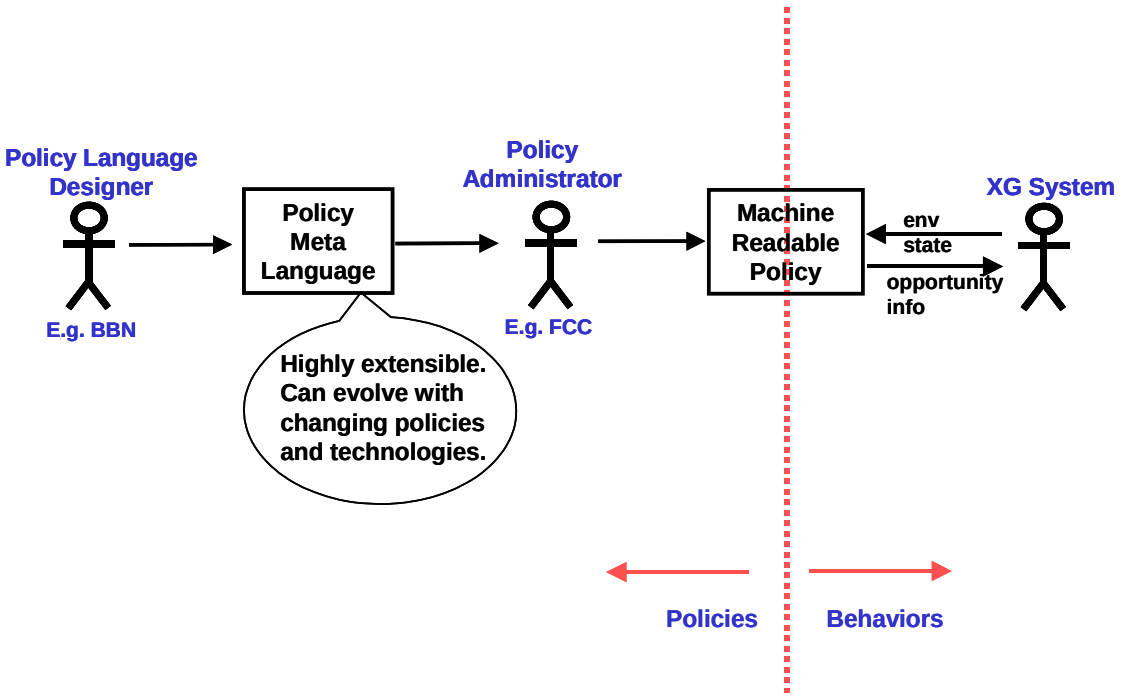


Figure 26: XG Policy Language Framework -- actors and roles

The *policy language designer* creates the language model that defines the high-level objects of the language along with the language syntax and semantics and publishes it (e.g. at a well-known URI) for the policy designers and policy users to access.

The *policy administrator* is responsible for developing and encoding spectrum policy as expressions in the policy language produced by the language designers. The policy administrator must then make the policy available to all spectrum users. The administrator does not have to know all the details of the language as they may use a graphical tool to encode the policy, called an instance editor, and hide the notational complexity of the language. The administrator also may also validate the policy set to ensure that there are no conflicts.

The *spectrum user* (an *XG system*), must then be able to use the policy to understand usage constraints (as specified by the policy administrators) on spectrum that may be available for its use. While the spectrum user needs to be able to understand the *policy* encoded in the language, the degree of language knowledge necessary to extract the policy information may vary greatly. The most limited interaction would be a *policy interface* that completely hides the language complexity, but may limit the type of information that may be extracted from the policy. The choice of the appropriate mechanism to extract information (from policy instances represented in an openly specified language representation) is therefore up to the system designer. If necessary, the policy may be transformed into a more compact format that an XG radio may use instead of the adopted standard language representation. An XG radio may also take advantage of the full power of the XG policy language and use policies encoded in it directly. In all cases, users must have the ability to access the policy and verify the XG system's conformance to the policy before using the spectrum.

For the reminder of this subsection we consider the development of the policy language. The first question is: Can it be done? After all, we are attempting to encode information that has traditionally been developed for *human* consumption. Further, there is a vast diversity in the primitive objects that make up regulatory policy domain – from concepts of frequencies to power spectral density, mathematical formulae, geographical concepts, time concepts including zoning, possible database access etc. Finally, it is not sufficient to be able to just *express* the information – it must be done in a manner that conveys the structural relationships amongst the objects so that a machine can *reason* about policies so that every single fact does not have to be encoded.

Although admittedly daunting, there are recent advances in a couple of fields that can be leveraged. First, the general area of *knowledge representation* has yielded tools, techniques and insight into representing human

consumable information. Second, there has been considerable research into languages and tools for the *semantic web*, in particular, markup languages that encode the semantic content along with the data that are also relevant for our purposes.

Another observation is that most (though certainly not all) of the spectrum policy domain is hierarchical in nature. The hierarchy stems from progressive narrowing along several dimensions: from general to specific, and along geographical regions. For instance, a general policy could be "not allowed to transmit on band B". This may be tightened as "not allowed to transmit in band B, except in frequencies *f1* and *f2*". Policies regarding general TV bands, HDTV band(s) and ATSC lend themselves to hierarchical nesting. Similarly, national policies may be "inherited" and made more specific in certain regions (e.g. U.S states). And devices with highly directional antennas may be able to operate under a specific (less restrictive) policy where similar devices with omni-directional antennas would need to operate under a more general (more restrictive) policy.

Finally, although the current policies are largely unstructured and disorganized from an information-scientific viewpoint – as would be expected given the lack of a formal structure before policies were written – this doesn't necessarily have to be the case with XG policies, which is the domain for our language. Establishing a framework and a language structure can help guide the specification of policies in a "cleaner" way.

These observations lead us to an approach that, at the highest level, could be termed a *structured declarative* language approach. It is declarative in the sense that the policies are expressed mainly in terms of *facts* and *rules*, and structured in the sense that the facts and rules are organized in an object-oriented fashion that exploits the power of inheritance to simplify the relationships.

Within this overall approach, it is instructive to dwell briefly on the requirements of our policy language. These include: *inheritance and polymorphism* – to enable policy rules and properties to extend others and reduce the need for enumeration; *reification (rules about rules)*, for example, to make a policy rule governing when or where a set of policies will apply; *inference*, to infer facts that may not be explicitly stated; *extensibility* in its vocabulary, structure and semantics so that the language can adapt to express new types of policies as spectrum policy requirements change; and finally, *standards based*, so that adoption is easy and tools continue to be available.

The use of policy language allows us to isolate the general framework within which XG needs to operate, from the details of establishing and advocating specific thresholds and rules of operation. We will develop this language so that we can demonstrate that it can support the types of operational controls that national regulatory authorities would wish to impose. Initially, conservative rule sets and thresholds may be imposed, but these can be broadened and tailored as experience is gained. These changes would not affect the design of the underlying XG systems, as the policies would be isolated from the XG implementing behaviors.

This approach allows policies to be written in advance of technology. It also allows for lower cost XG variants that do not have the capability to enforce certain conditions, and trade cost for potential performance. If a radio has a "better" FFT, then it can play closer to the edge. We avoid a single definition of XG functionality – rather, we embrace a set of design choices that implementers can match to spectrum and policy conditions.

3.2 Abstract Behaviors

When considering policy-based control of an XG radio, we believe that control must be limited to *what* the radio does, rather than *how* the radio does it. For instance, it makes sense for a policy to say that the transmit power must be no more than 10 mW, but mandating that the radio use a particular automatic gain control (AGC) mechanism or even use AGC does not seem appropriate. In order to capture this in our framework, it is essential that we decouple the *how* from the *what* – this naturally leads to the concept of *abstract behaviors* as distinct from *protocols* that implement such behavior.

An abstract behavior is an abstraction of a mechanism that hides details of one or more aspects of its functionality. For instance, an "interference limiting" abstract behavior could be for a node to "not introduce a signal that increases the interference over any point beyond 400 meters of the node by more than 8%". As a

further example, consider that TCP is a *mechanism* which implements the transport layer *functionality* defined by the OSI reference model. TCP exhibits two *abstract behaviors*: setting up, and maintaining a connection between networked applications.

The abstraction of behaviors could be done at several levels, and so a particular protocol may correspond to several abstract behaviors. For instance, consider the IEEE 802.11 MAC Distributed Coordinated Function. A protocol for this involves specifying the frames (RTS/CTS/DATA/ACK), their formats (waveforms), timers, finite state machines, and so on. An abstract behavior might be to simply say "… use RTS/CTS/DATA/ACK handshake for collision avoidance…". This behavior might be implemented by a variety of protocols that might differ in packet format or how the NAV is handled. An even higher level abstraction might be to say "… must avoid collisions…" allowing different kinds of algorithms, including TDMA. For XG, we will choose appropriate levels on a per protocol basis, based on standardization and regulatory considerations.

An XG system is comprised of a number of mechanisms working in unison. Each of these mechanisms may be seen as solving a particular problem, which typically has more than one solution, and correspondingly, more than one protocol. Each solution has its own advantages and disadvantages, and may be appropriate for different scenarios. However, it is very difficult to regulate every possible mechanism. On the other hand, mandating a select few mechanisms stifles innovation.

How then can we allow innovation, and a variety of protocols, while assuring regulatory conformance? We propose to accomplish this by specifying a *core set* of abstract behaviors. Each mechanism within the XG system can be thought of as composed of two parts: one part that falls within regulatory purview, and one that doesn't. Typically the part that falls within regulatory purview is likely to be small and the one that doesn't is likely to be large and, moreover, is the one in which much of the scope for innovation lies. We envision that a behavior from the core set of specified abstract behaviors will be associated with each mechanism as the regulatable part, while preserving the freedom for the mechanism to make optimizations as desired in the unregulated part. Specifically, we shall first establish a set of behaviors that ensure systems are *interference limiting*.

We shall develop a framework that assures that optimizing methods lie outside of the regulatable kernel, as this will enable the continued progression of XG capability and performance, without requiring that these actions be addressed within a regulatory process. A generalized framework is optimal for DoD interests, as it provides a means to address a large number of systems within a single context – a context that may well have been adopted to enable civil uses as much, or more, than military ones.

This subset or the "core set" within the boundary will be referred to as the *regulatable kernel*. It is illustrated in figure 27, which shows four abstract behaviors A,B,C,D, and three mechanisms that incorporate these behaviors. Those portions of the XG system that are within the core set (indicated by the thick circle) should be of interest to the regulatory community, as these contain the functions that achieve the objectives subject to regulation.

By isolating this subset of the XG system, we hope to provide a focus for regulatory consideration that is compact, and does not necessitate the regulatory community becoming involved in all aspects of XG implementations.

This approach is similar to the way security systems work today. In the security realm, we have adopted methods that isolate the critical aspects of the larger system into a small, and highly controlled subset, sometimes referred to as the "trusted security kernel". Typically this provides a Trusted Kernel, Physical Red/Black isolation, cryptographic or a similar mechanism that enables us to think of the bulk of the system as outside of the security boundary. As long as we know, prove, and trust the mechanisms within this trusted kernel, other parts of the security system can evolve to meet different cost-performance tradeoffs. We need a similar framework for XG.

Mechanisms outside of the kernel may extend the functionality to apply it to specific situations. For instance, a regulatory kernel behavior might be to dictate an upper limit to the field intensity at any point (omnidirectionally) beyond some specified distance from the transmitter. Outside the regulatory kernel would be a behavior that assures the limit is met at each environmental system by recognizing a capability for the transmitter to know where each environmental system is and to limit the transmission in that direction so as to meet the regulatory kernel requirement at that point. This "directional sensing and control" behavior would be an extension of the "interference limiting" behavior beyond the Regulatable Kernel.

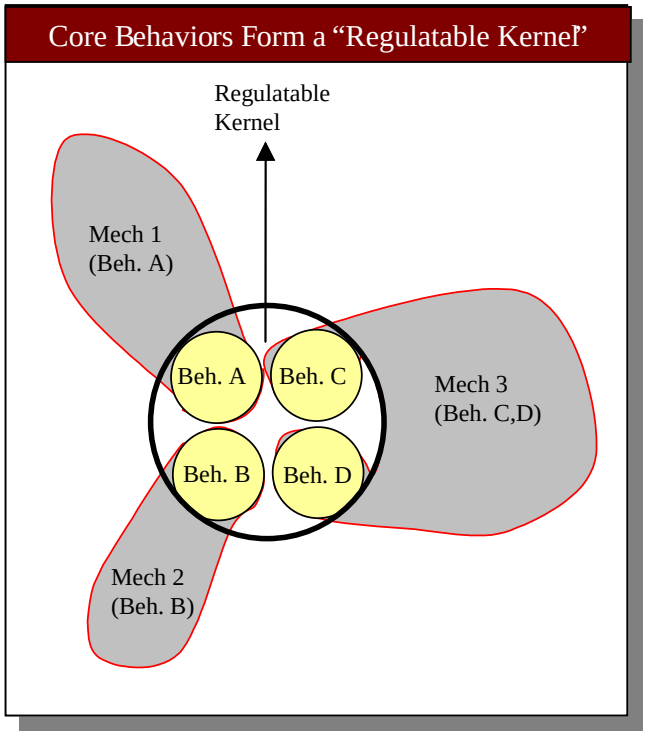


Figure 27: Core Behaviors

There are a number of challenges to be addressed in the realization of this vision. First, we need to identify a suitable set of abstract behaviors. If there are too few, it will restrict the scope of XG mechanisms. On the other hand, if there are too many behaviors, the regulatory process might become too complex. A guiding principle is to keep as little as possible within the core behaviors, subject to traceability of policies into those behaviors. Second, the level of detail of each abstract behavior must be determined. If it is too detailed, it might be difficult to obtain regulatory approval and to achieve consensus between two XG implementers. If it is too general, an approver may need to look at the implementation in order to determine whether or not it is interference preserving. Further, it will increase the risks of producing non-interoperable implementations.

Specific instantiations of abstract behaviors will permit interoperability among XG systems, and the abstract behaviors allow for fine grained optimizations within a group of XG nodes. However, correct definition and a suitably clean partitioning between core/non-core behaviors rather than interoperability is the primary goal of abstract behaviors.

The need for progressive layers of abstraction, the need to hide details, and the need to allow multiple ways of implementing a particular defined behavior strongly suggest the use of an object-oriented approach. Object-oriented design and specification are increasingly being recognized as the cleanest and most efficient way of system design and development, and use of this approach can benefit the development of XG.

In our object-oriented analysis framework, the basic XG classes have defined interfaces and high-level behaviors, and the details are left for instantiation by the behavior implementation. For example, there is an XG class that manages information about spectrum occupancy. Regardless of its implementation, we will require that it have access methods that address energy and frequency. Different implementations may have different noise floors, frequency resolution, scan rate, etc. The interference methods will need to understand how these parameters affect the ability to determine non-interfering opportunities, but need not be specific to these values.

Similarly, we can describe an abstraction of the process of using a set of spectrum occupancies to select a spectrum opportunity for a given spectral power density, time, bandwidth, etc. Actual implementations will vary, but must be constrained to ensure compliance with the policy-based language controls. So long as the XG

operation can ensure this compliance, we will leave the implementers free to develop and evolve ever more capable instantiations of these behaviors.

In many cases, the behaviors have a natural hierarchical structure. Using an object oriented framework, this structure can be elegantly represented by means of "inheritance" of classes. For instance, consider the process of identifying a set of opportunities. This may involve using only local sensing information ("uncoordinated") or be based on the sensing information from neighboring nodes ("coordinated"). The latter would be useful, for instance, if a deep fade is the reason that local sensing shows up an opportunity and in reality the opportunity is not there. For each case, one may augment the process by adding "probe" transmissions to perhaps trigger a response from a primary receiver ("active sensing"). With an object oriented framework, the top level class would be inherited in the requisite way to specify the different solutions to the same problem.

An example set of core abstract behaviors is as follows. Note that each is a class of behaviors that can be inherited and instantiated in a number of ways – details are given in the XG Abstract Behaviors RFC.

1. The *XG Spectrum Awareness Management (XG-SAM)* behavior, which describes how opportunity information is acquired, identified, represented and disseminated within and across XG systems. XG-SAM encompasses awareness information gained from sensing, policy (including configuration), and through XG Opportunity Dissemination Protocol (XG-ODP) instances, as well as opportunity identification.

2. The *XG Opportunity Dissemination Protocol (XG-ODP)* behavior, which is a class of protocol behaviors that can be used by XG systems to share opportunity awareness information. An XG system should participate in one or more instances of XG-ODP classes.

3. The *XG Usage Accounting Management (XG-UAM)* behavior, which enables every emission to be traced to a valid opportunity and a set of policy rules that allows this usage. Therefore each emission must be tagged with an opportunity object and a policy object.

3. The *XG Use Coordination Protocol (XG-UCP)* behavior, which allows XG systems to coordinate the use of selected opportunities with other (XG and non-XG) systems. XG systems should participate in one or more instances of one or more XG-UCP classes.

4 XG Operation

In this section we discuss interaction between different XG networks, layering issues involved in the dissemination of XG information, operational modes, and other concepts involved in the operation of an XG network.

Consider an XG node that enters a geographical area populated by other spectrum users. There are several possibilities for such users: they may be licensed "primaries" with right-of-use, or unlicensed. If they are unlicensed, they may or may not be XG (or more generally, opportunistic spectrum access capable) nodes. And if they are, they may or may not be able to interact with the XG network in question. Finally, there may be several levels of interaction – from just deconflicting to complete participation in spectrum allocation and use.

The following definitions are helpful in discussing such interactions. From the perspective an XG node *X*, there are three kinds of "other" nodes:

- *Non-XG*. These are "traditional" non-XG-capable nodes, or are running a different (incompatible) set of protocols. Node *X* will be able to sense their radio energy, and possibly identify physical layer waveforms (with appropriate feature detectors), but that is the limit of the interaction.
- *XG-aware.* These are nodes that implement some common protocol—that may or may not be part of the XG suite of protocols – that enables the exchange of spectrum usage information that allows deconflicting. That is, for instance, node *X* can determine which opportunities are being used by such nodes and "stay away" from those. Depending upon the nature of the common protocol, the exchange may be one-way or two-way – for example, node *X* may be able to determine the opportunities used by an XG-aware node *Y*, but

not vice-versa. However, they do not cooperate in terms of opportunity allocation/assignment, and are not required to know about which opportunities are sensed/used by each other.

- *XG-cooperative.* These are nodes with which our protocols can coordinate the use of spectrum. That is, XG-cooperative nodes implement a set of protocols that allow them to exchange opportunity information, perhaps over multiple hops of XG-cooperative nodes, and to negotiate allocation of opportunities. In other words, XG cooperative nodes sense opportunities, disseminate and receive such opportunity information from other nodes, and jointly allocate such opportunities. We also simply call these *XG nodes* and a collection of XG nodes an *XG network*.

Orthogonal to the above categories is the classification into *licensed* and *unlicensed*. Licensed nodes have right-of-use and node *X* must be interference limiting with them. Unlicensed nodes may not have such protection but node *X* is likely governed by policy that requires "good citizenship" in the sharing (a "commons" model). Thus, in our vision, there are 6 modes of possible interactions that XG will accommodate.

We now consider the *scope* of interaction between XG nodes. We first define the concept of an *XG domain*, which influences the nature and scope of our protocols. An initial vision of an XG domain is a collection of XG-cooperative nodes that form a connected (at layer 1) network. That is, there is a layer 1 path (could be multi-segment) from every node in the XG domain to every other, and each node in that path implements a common protocol for awareness, dissemination and use of opportunities.

This concept is illustrated in Figure 28. The straight lines indicate connectivity between nodes. Note that the concept of a subnet (a layer 3a functionality) is independent of the XG network. Indeed, if XG nodes from two different layer 3 subnets are within range of each other, they can belong to the same domain (beneath layer 3) crossing the subnet boundary, as illustrated in the figure. What this means is that an XG node in one subnet may know the frequencies used by an XG node in another subnet if there is a path through other XG nodes from one to the other.

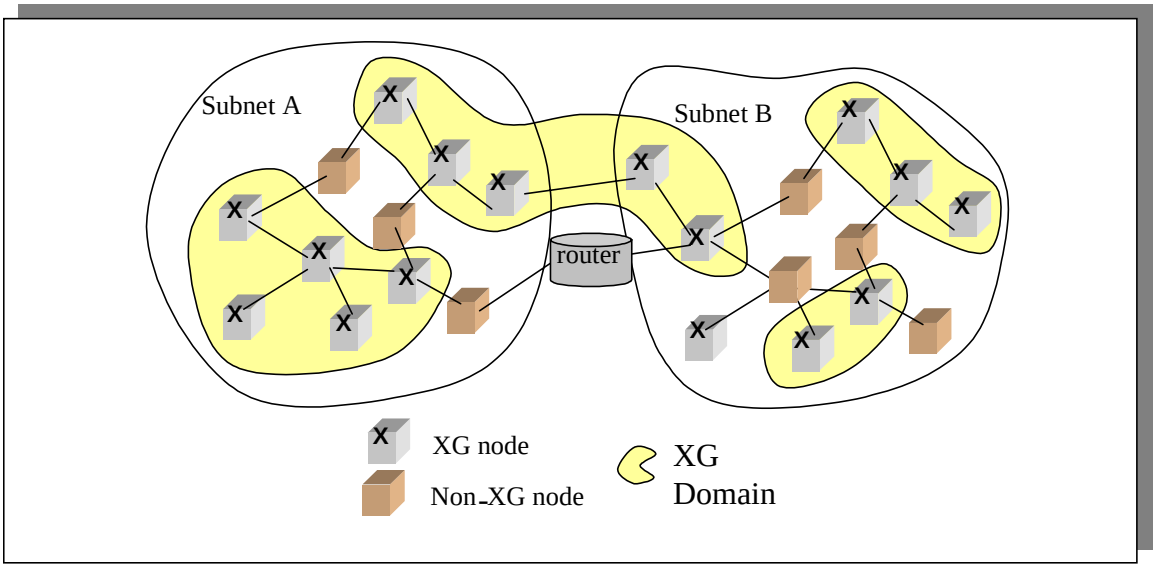


Figure 28: Concept of an XG domain. A domain is the set of nodes within a shaded region. A line between two nodes indicates the existence of a physical layer connection between the nodes. Note that network-layer connectivity may be different, and is not shown.

In this vision, an XG domain is not a packet or data centric network, but one that operates as a Layer 2 information fusion and dissemination structure. Several instantiations of this general idea are possible – from simply relaying or "flooding" the opportunity information to all nodes in a domain, to merely maintain awareness of certain content (such as spectrum usage at their location), some neighboring nodes, and local frequency planning. In general, information may be provided only on a "need to know" basis with other nodes, but

reprocessing it only to provide the minimal set of awareness. For instance, information can be correlated so that each node does not report common blocked frequencies, and only candidate frequencies are relayed. This can both eliminate significant overhead bandwidth, while ensuring that enough information is available for the decision process at each node. Exchange of opportunity information across a domain should not be confused with layer 3 "routing"even if flooding is used – we are merely talking about layer 2 exchange of control packets beyond a single-hop.

This framework does not require routing information to be distributed, as each node joins only to other nodes that are within the same RF environment. No XG unique address management is needed.

How are nodes in the domain organized? Broadly speaking, two opportunistic spectrum access usage models are possible, as briefly described below:

- *Centralized*. In this approach, the management of spectrum opportunities is controlled by a single entity or node, called the *bandwidth manager*, or *spectrum broker*. This node is responsible for sensing, and deciding which opportunities can be used, and by which nodes. This is relevant in real-time secondary spectrum markets.
- *Distributed*. In this approach, the interaction is "peer to peer". In other words, the XG nodes are collectively responsible for sensing and sharing the opportunities. This is most relevant in military networks.

There can also be models that share properties of both these types. Another spectrum sharing model that is related but does not quite fit in the broad classification is the "interruptible" model. In this model, spectrum is allocated subject to revocation by the primary holder of the spectrum. One may think of this as a special case of the more general XG model – there is either no primary or a primary all the time (recognized by, say, a signature "get out, I need it right away" continuous signal). Interruptible spectrum may be seen as a policy issue where the requirements for using the spectrum are very stringent.

Our XG approach is applicable to both centralized and distributed architectures. Indeed, policy may decide whether or not the system functions in a centralized or a distributed manner.

Within each approach, a number of possibilities exist. An example taxonomy is shown in Figure 29. There are three possible implementations of the centralized model: 1) there is no sensing, and the band manager has a block of spectrum that it owns that is given to secondary users in real-time based on requests; 2) only the band manager senses the spectrum, identifies opportunities dynamically and allocates it to XG nodes; 3) like (2), but additionally, XG nodes sense and provide information to the band manager to help in decisions.

The XG framework should accommodate all of these possibilities. Some of these, such as the spectrum handout approach are a special case of the general one – in this case, there is a "null" sensing and identification behavior.

5 Summary

The current policy of statically assigning spectrum for services can be inefficient in terms of spectrum utilization and cumbersome in terms of deployment agility. Opportunistic spectrum access, that is, the idea of opportunistically using assigned spectrum in an interference-limiting manner holds great promise.

The XG program is developing the concepts, framework and enabling technologies for opportunistic spectrum access. Specifically, the program has two goals: develop the enabling technologies for opportunistic spectrum access, including solutions to the problem of sensing, characterizing, identifying, distributing, and allocating spectrum opportunities; and develop a long-lived framework for managing key aspects of radio behavior through flexible application of policies. Our vision encompasses not only spectrum agility, but also policy agility – that is, the use of machine understandable policies for controlling the behavior of an XG radio.

The XG framework decouples policies, behaviors and protocols. The decoupling is enabled using two key concepts: the use of a policy language, and the definition of (core) abstract behaviors.

Spectrum policies are expressed using policy scripts based on an XG policy meta language that the XG program will define. By having policy scripts tailored to reflect national and regional considerations, considerable control can be exercised over the behavior of the XG system. Traceability of policies to behaviors is an important goal of the XG framework and will help the accreditation process.

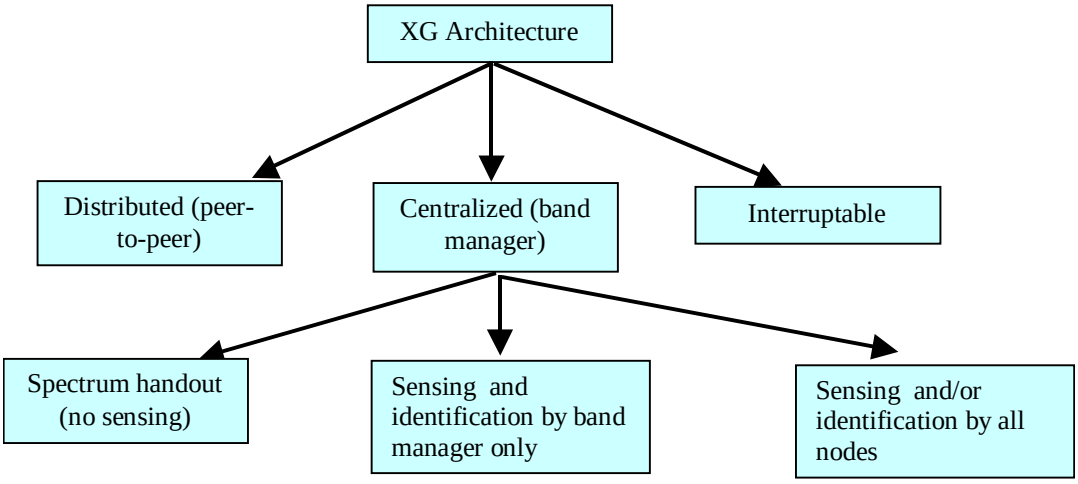


Figure 29: Taxonomy of XG approaches

A core set of interference limiting abstract behaviors will be defined. This core set is the necessary and sufficient set of mechanisms for regulatory approval. Optimizations are best left to commercial innovation arising from competition. By separating the innovations from the core set of behaviors, we enable the continued progression of XG capability and performance, without requiring that these actions be addressed within a regulatory process. The specification of the behaviors will be done at several levels of abstraction, using an object-oriented approach.

The XG vision encompasses more than a point solution to the problem of dynamic spectrum sharing. Rather, its goal is to develop a framework for diverse solutions to co-exist while sharing a core set of behaviors.

A number of network interaction and organization possibilities and spectrum management exist within the XG context. XG will accommodate interactions with non-XG, XG-aware (deconflicting use), and XG-cooperative (coordinating use) nodes. Organization models include the centralized "band manager", distributed "peer-to-peer", interruptible and others. Our goal is to accommodate as many of these diverse implementation as reasonable within the policy-centric and behavior-oriented framework.

In sum, our vision is to enable two new regimes: a new spectrum access regime consisting of technologies that sense, characterize, and utilize spectrum opportunities in an interference-limiting manner; and a new regulatory control regime consisting of methods and technologies for controlling such opportunistic spectrum access in a highly flexible, traceable manner using machine understandable policies. We shall enable the latter by defining an XG framework whose key components include the definition of a policy language and the definition of abstract behaviors.

Acknowledgments

This document was prepared by the Internetwork Research Department, BBN Technologies, with input from members of the XG working group.

Comments

Comments on this RFC should be emailed to xg-rfc-comments@bbn.com, with a carbon copy to Ram Ramanathan at ramanath@bbn.com, along with the commenter's name and organization.

Appendix B
XG Architectural Framework
Request for Comments

Version 1.0

1 Purpose

This document describes the architectural framework for the development of XG. It also summarizes the main features of XG protocols, interfaces and policies, each of which are detailed in separate documents. This document may be thought of as the "overview" document for the XG protocols. Specifically, the following are addressed:

- Requirements of XG protocols
- Where and how XG functionality fits within a typical networking system
- Layering issues
- Functional decomposition into modules, and interfaces
- Summary of protocols, APIs and policies

This document is a Request for Comments (RFC). Accordingly, an important purpose of this document is to obtain feedback from the community at large, and to refine the ideas described here based on that feedback. The development of the XG architectural framework is an evolutionary process, and this document reflects a snapshot in that evolution.

A number of other RFCs related to XG exist, or are being planned. The complete XG family will consist of the following:

1. *XG Vision RFC*. This lays out the motivation for XG and its scope, presents the key concepts underlying XG, and describes an approach for defining XG.
2. *XG Architectural Framework (AF) RFC*. This document.
3. *XG Protocol RFCs*. Each XG protocol RFC specifies an abstract behavior, and when appropriate, details of individual protocols.
4. *XG Policy RFC*. The XG policy RFC describes the syntax of a policy specification meta-language and a set of example policy specifications.
5. *XG Interface RFCs*. We currently envisage two interface RFCs – XG Transceiver API and XG Opportunity API. These detail the primitives corresponding to these interfaces.

We recommend that this document be read after the XG Vision RFC and before any of the other RFCs.

An important purpose of this document is to articulate, at the highest level, a viable solution that addresses the needs outlined in the XG Vision RFC. In turn, this document is the blueprint to be followed by the protocol, policy and interface RFCs, each of which details a particular aspect of the architecture.

There are, of course, several ways of approaching and defining an XG architecture that realizes the XG vision. Furthermore, the near-term needs are somewhat different from the longer-term needs. In the near-term, the emphasis should be on simplicity and in the longer-term on completeness and lasting value. Therefore, in this document, we have outlined two architectures – near-term and long-term. These will be refined in parallel and merged at a future date – that is, the near-term architecture will evolve and be subsumed within the longer-term architecture to result in a single architecture going forward.

2 Requirements

The design of any system should be guided by a clear set of requirements. This section identifies requirements for the XG architecture and protocols. Every aspect of the architecture and protocols should be traceable to one of these requirements, and eventually, every requirement should be supported by some aspect of the protocols.

Requirements include (we don't include obvious requirements, such as "must provide dynamic spectrum access with interference mitigation", which follow directly from the XG vision).

1. It must be possible to add XG to a legacy system. Such an addition should not require extensive modifications to the legacy MAC mechanisms.
2. Legacy systems without XG extensions should interoperate with XG-enhanced systems.
3. There must be a provision to incorporate spectrum policies, priorities, and exclusions into the functioning of the protocols and/or abstract behaviors.
4. The XG protocols must be largely agnostic to the MAC layer technologies. They must not depend upon how a MAC layer functionality is implemented. The general behavior of an XG system should be largely independent of the nature of the MAC layer, though a particular XG implementation may be aware of and exploit particular MAC layer technologies.
5. The XG protocols must be largely agnostic to the physical layer technologies. That is, it must not depend upon how a physical layer functionality is implemented. The general behavior of an existing XG system should be largely independent of the nature of the physical layer. A particular XG implementation may be aware of and exploit particular physical layer technologies.
6. A core set of behaviors must be identified in such a manner that a viable architecture where only the core set needs to be considered for regulatory approval is possible.
7. The framework and protocols should be flexible enough to support XG-like capabilities long after the initial DARPA XG implementation(s).

A major goal of the RFCs is to present an *abstract* view of XG. In particular, the problem statement is not with respect to any one existing protocol, nor will the solution be simply an embellishment of an existing protocol, such as 802.11. While it is likely that such embellishments will prove useful for the initial implementations of XG, the RFCs themselves will be at "one level higher" and solve the *generic* XG problem.

Another goal is to keep the core behaviors distinct from the innovations that may implement the mechanisms in different ways. This would be analogous to secure kernels – that is, inside the boundary, we can be sure of what is happening and can trust it whereas outside this boundary there is room for innovation. The challenge is to make it so that only the core set of behaviors "inside the boundary" is relevant for regulatory approval.

3 Preliminaries

The description of the XG architecture will employ several concepts, such as layers, modules, interfaces, behaviors, and interference and XG domains. In this section, we define these concepts precisely so that all readers may interpret the rest of the document uniformly.

3.1 Layering

A *layer* is a level of abstraction that captures some important aspect of the system, provides an interface that can be manipulated by other components of the system, and hides the details of how the encapsulated functionality is implemented. A *protocol* is a set of rules governing the format and meaning of messages that are exchanged by the peer entities at the same layer. A protocol provides a communication service that higher-layer objects use to exchange messages. The basic idea of *layered protocols* is for a lower layer to provide services to the layer above it.

A layer can be divided into smaller logical substructures called *sublayers*. Sublayers abstract functionality within a layer in just the same way that layers abstract functionality within a system. Clearly, this definition can be generalized recursively.

When two adjacent layers (or sublayers) differ in some aspect of functional perspective, and we need to make the differences transparent to the higher layer (or sublayer), the concept of a *convergence layer* or "adaptation" layer is useful. A classic example is the layering of IP over ATM, where the variable-length IP packets need to be segmented into fixed length ATM cells. A convergence layer is introduced that does segmentation and reassembly. The convergence layer is similar to a layer, but is used when a simple, specific functionality is targeted in relation to "mapping" between two adjacent layers. In other words, there is no new significant functionality introduced, but a kind of "translation" happens. In our case, the diversity of technologies at the physical and MAC layers and the need to map between them motivates the use of convergence layers.

3.2 Modules and Interfaces

Layers, sublayers, and convergence layers are connected by *service interfaces,* or *APIs* (we use the latter term). An API provides a set of *primitives* using which the service provided by a (convergence/sub)layer can be suitably abstracted. Use of APIs provides a means for modular development of system components, perhaps by different performers. APIs between layers/sublayers are termed *vertical APIs*. In contrast, *horizontal APIs* are interfaces between modules at the same layer.

3.3 Domains and Regions

The application of XG principles and techniques is concerned with maintaining predictable levels of interference among potentially competing radio systems. We distinguish between the radio systems themselves and the locations in space that they occupy with the following definitions.

The terms *set* and *domain* refer to collections of individual radio systems (sometimes called *nodes*). The term *region* refers to a geographic area in which one or more nodes may be located. There are different types of sets, domains and regions.

The i*nterference set* of a node consists of all nodes with whom that node may interfere. An *interference domain* is the transitive closure of the interference sets of one or more nodes.

An *interference region* is the contiguous area occupied by an interference domain, extended to include the area that would be occupied by "would-be" interferers with members of that domain. That is, the area subject to the interference of/by members of the interference domain.

An *XG domain*, on the other hand, refers to a collection of nodes that are able to exchange opportunity information for the purpose of making choices about the use of spectrum or other transmission-related resources. Members of an XG domain are able to cooperate in the utilization of spectrum-related resources.

3.4 Channels and Opportunities

We assume that the operational spectrum for XG can be partitioned into non-overlapping *channels*, which is the fundamental "unit" of spectrum for dynamic management. For instance, a 100 MHz band could be partitioned, for XG purposes, into 10000 channels of 10 KHz each. It is not necessary to have channels of equal width. Properties (such as presence of a primary signal) are determined on a per channel basis. A channel is the smallest unit for which such properties can be described.

An *opportunity* exists if an XG node can transmit using some combination of operating parameters such that existing primary nodes (if any) do not perceive interference, for a given threshold of such interference. We note that the definition of the opportunity is node and threshold dependent (amongst other things), and so the same "sensing" information may or may not represent an opportunity.

3.5 XG Nodes

The XG nodes and the networking context were discussed in detail in the Vision RFC. Here we reiterate the terminology for the different kinds of nodes that play a role in the architecture.

- *Non-XG.* These are "traditional" non-XG-capable nodes, or are running a different (incompatible) set of protocols. Our XG protocols must be interference preserving with respect to such nodes under the assumption that they are operating legally.
- *XG-aware.* These are nodes that can exchange information about what frequencies they are using and may make use of information about the frequencies that our node/network is using. However, they do not cooperate in terms of frequency assignment. Our XG protocols must find out and avoid frequencies used by such nodes, and should inform them about the frequencies we use.
- *XG-cooperative.* These are nodes that can use the XG protocols to coordinate the use of spectrum. They run an interoperable implementation of the XG protocols. These are nodes with which distributed dynamic spectrum sharing typically works. We often call these simply *XG nodes.*

3.6 Abstract Behaviors

Finally, we discuss the concept of *abstract behaviors*. An abstract behavior is an abstraction of a protocol that hides details of one or more aspects of its functionality. This hiding could be done at several levels, and so one could have several levels of abstract behaviors. For instance, consider the IEEE 802.11 MAC Distributed Coordinated Function. A protocol for this involves specifying the frames (RTS/CTS/DATA/ACK), their formats (waveforms), timers, finite state machines, and so on. A first level abstract behavior might be to simply say "… must use RTS/CTS/DATA/ACK handshake for collision avoidance…". This behavior might be implemented by a variety of protocols that might differ in packet format or how the NAV is handled. An even higher level abstraction might be to say "… must avoid collisions…" allowing different kinds of algorithms, including TDMA. For XG, we will choose appropriate levels on a per protocol basis, based on standardization and regulatory issues.

4 XG Framework

The XG system is a complex one. Architecting complex systems has long been recognized as a problem that is best addressed using a formal framework. One of the approaches that has become popular is the Zachman Framework for Enterprise Architecture. In this section, we adapt the ideas behind the Zachman Framework to present an XG framework. Such a framework is helpful in providing a panoramic view of XG that puts in perspective the material presented in this and other RFCs.

4.1 The Zachman Framework

The key idea behind the Zachman Framework is that a complex system may be viewed at different levels, by different "players". The classic example, and one that was introduced in the original paper by Zachman [Zach], refers to the construction of a house. In this, the owner often gives broad requirements for the house; the architect prepares architect's drawings that depict the house from the owner's perspective. These are then made into formal plans that are a designer's representation of the final product which would be used by the contractor, who in turn makes detailed engineering plans to be used by the builder, and so on. Each of these levels is important and used by a particular person in the development process.

The crux of the Zachman framework is a matrix where each row represents a different *perspective* of the system. These views include contextual (scope), conceptual (business model), logical (system model), physical (technology model), and detailed representations. Each column represents a different *description* of the system. Descriptions include what (data), how (function), where (network), who (people), when (time), and why (motivation). Thus, each cell in the matrix captures a unique aspect of the system, and is explictly differentiable from all other cells in the matrix.

The Zachman Framework does not indicate a methodology for filling out the cells, nor does it offer any guidelines on what should be done with the matrix. Rather, it appears to be a tool for enhancing clarity of thought, and as a structure that ensures that all aspects of the system are covered.

4.2 XG Framework

Like many other complex system, XG also needs to be understood at many "levels", from conceptual to detailed implementation. Each "player" in the development of the XG system has a different view of XG, according to his or her requirements. For instance, the FCC has a view that is largely focused on regulatory issues and on ensuring interference preservation, which is quite different from the system integrators view which needs to consider performance and other aspects. All of these different views are important. It is helpful at this stage to consider the broadest picture of XG and look at these different views, so that each RFC, including this one can be put in perspective.

The Zachman Framework was designed for an enterprise, and XG is not an enterprise. Thus, it is not directly applicable for our purposes. However, the key idea behind the framework – namely, that there are different perspectives and descriptions – may be taken and adapted to our context. Such an adaptation is presented in Table 4 below. We have changed the semantics of the rows slightly and considered only the three most relevant items in the descriptions.

The XG program benefits from such an architectural framework because the operational views provide a means for communicating with high-level decision makers, e.g., DoD, commercial and FCC executives. The conceptual and design views allow the design of core behaviors that can be used by the high-level decision makers for regulatory purposes. The system views allow engineers to understand the XG system functions and interfaces, and the technical views show how XG fits into the existing standards and how it adapts to changing standards in the future.

	WHAT	WHY	HOW
Scope (DoD/FCC)	♦ Dynamic spectrum management	♦ Increased capacity ♦ Better use of spectrum ♦ Zero setup time ♦ Regulatory simplicity	♦ Build a system that uses unused frequencies in an interference preserving way
Concept (XG PM)	♦ Abstract (core) behaviors ♦ Protocols ♦ Policy language	♦ Long-term impact ♦ Regulatory approval ♦ Flexibility	♦ RFC process ♦ Performer participation ♦ Industry feedback ♦ FCC/DSO involvement
Design (Contractor: BBN and working group)	♦ Architectural framework (near and far). ♦ Behavior specs – sensing, identification, allocation, use.	♦ Long-term impact ♦ Regulatory approval ♦ Flexibility	♦ AF RFC ♦ Behavior RFCs ♦ Policy RFC ♦ API RFC ♦ WG interaction
Technology (Contractors: SS, Raytheon, LM)	♦ Sensing algorithms ♦ Transmit power estimation behaviors ♦ Allocation algorithms ♦ Morphed waveforms ♦ Tieing up with existing protocols (e.g 802.11) ♦ Component technology	♦ Interference preservation ♦ Noise temperature control ♦ Backward compatibility with legacy ♦ Future upgrades in a competitive manner	PROPREITARY

Table 4

5 XG Layering Issues

In our vision, XG is implemented as a layer 2 process as shown in Figure 30. A legacy stack (left) may be upgraded to use XG without modification to the legacy MAC protocol. The legacy Transceiver API

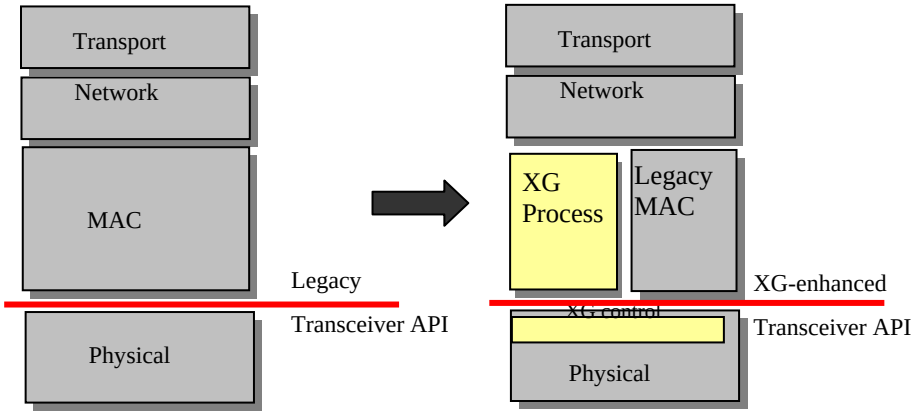


Figure 30 - XG Process in Stack Context

may be enhanced to include certain XG-specific primitives to provide an XG-enhanced Transceiver API as shown in the figure (right). Note however that this does not require a change in the legacy MAC protocol as it continues to use the subset of the API that it originally used. Thus, the legacy MAC need not be aware of XG. There is no change to the network layer and above – the scope of XG is entirely restricted to physical and MAC layers.

The physical layer implements a minimal "XG control", in that it recognizes that some of the MAC requests may imply XG-specific action. The XG process communicates with peer XG processes at other nodes to exchange spectrum information, and other XG control information (Figure 31).

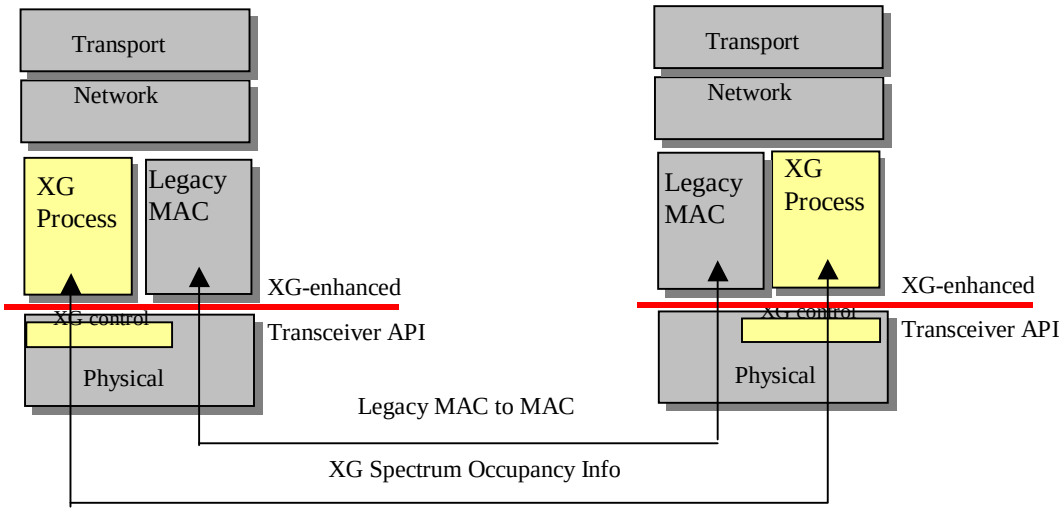


Figure 31: XG peer interaction

We note that the XG process being at layer 2 is with respect to *application data* packets. This does not preclude XG *control messages* from reusing some of the higher layer functionality – for instance, using TCP to communicate to a peer XG process in the same domain. We do not consider this a layer violation since the layering of functionality applies to data packets[8]. Our stack model does not imply that routing, transport, encryption has to be re-implemented at layer 2.

The XG processes coordinate with each other to implement a dynamic spectrum sharing procedure amongst themselves in a manner that is designed to control interference to existing primary users. The XG process then creates the "state" in the physical layer for appropriate handling of packets consistent with decisions made in the XG process. This is achieved by certain control primitives implemented as part of the XG enhancement of the Transceiver API. For example, the XG process might have determined that frequency channels *f1* through *f2* may be used by this node at this time. MAC packets then are transmitted on these channels. The state may also contain instructions on any modifications that may need to be made to the outgoing and/or incoming packets on behalf of XG. In other words, the XG control at the physical layer executes data packet processing on behalf of the layer 2 XG process[9].

The XG process utilizes the physical layer to communicate and exchange spectrum utilization perceptions, and then to coordinate frequency assignments for the radios in the physical network. This exchange is essential because we need to both ensure that the selected frequency is usable at the receiver, and is not likely to jam signals from the environment of the transmitter.

[8] This is similar to the use of TCP for control message delivery by BGP (Border Gateway Protocol) which is considered to be a layer 3 protocol.
[9] This avoids having to actually pass the data packets "up the stack" to the XG process – doing so is a "layer violation".

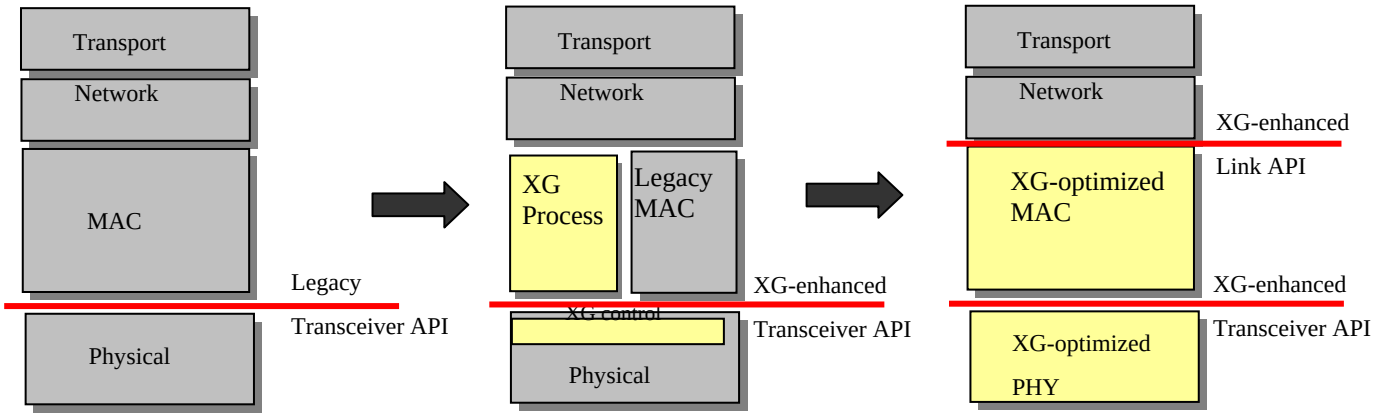


Figure 32 - Evolution from Legacy, through co-existing XG/Legacy, to Optimized XG

This approach has the advantage that it allows the use of XG with existing MAC designs. In other words, it should be capable of supporting legacy MAC code before the transition to XG-Optimized MAC and Physical layers is made. The transition from current day practice to "full XG" is shown in Figure 32.

In some cases, it is likely that the XG protocols cannot use the native physical layer. This is true when this layer has unique characteristics, or when there is no common mode of operation, such as in linking heterogeneous networks. It is also true in cases where the XG protocols require an appropriately XG-enhanced physical layer. In the most extreme case, one of the XG-enabled systems may not be a communications system at all, but may be a sensor that communicates and/or coordinates spectrum usage with communications or other sensor systems. In such cases, we perceive that we will need to define a physical layer standard for an XG interoperability path, which can be selected as part of the XG standard, or negotiated among the radios. Since Software Defined Radios are the likely implementation platforms for XG, the introduction of an additional physical layer is not as significant as it would be with discrete implementations, but is still a complexity we would like to avoid. Some means of determining a common mode of operation could be a more suitable solution. This is a technology that DARPA is investigating in other programs, and may remain outside of the current XG work.

The above representation is only the simplest form of XG. Clearly there are very significant benefits to the system's ability to be aware of, and to utilize network topology information that is only accessible in the upper layers, such as the membership data that likely resides in the network layer. We will be investigating these, and similar, opportunities for enhanced performance later in the program. We intend, if possible, to develop this functionality in the context of the same set of abstract behaviors that are used in the core architecture. We envision that with the above mentioned and other features, an "ultimate" XG architecture would have XG optimized MAC and physical layers with the network layer being made aware of XG features by means of an API (which could be used to supply network topology information, for instance). This is illustrated in the rightmost diagram in Figure 32.

6 XG Modules

In this section, we present a first-level modular decomposition of XG functionality. We begin by recalling the broad architectural vision presented in the companion document "XG Vision RFC", and then work our way through a decomposition.

We shall use the rightmost diagram in Figure 32 as a starting point for the decomposition as it is the most general, and also allows us to focus on the XG functionality. Note that this choice does not in any

way detract from the vision of having the legacy MAC and XG MAC coexist without changes to the legacy system. To see this, simply imagine that a legacy MAC box is placed alongside the XG-optimized MAC box in the rightmost diagram, and note that the XG-enhanced Transceiver API contains all of the primitives that the legacy system used (since it is an "enhancement"). Now, having affirmed this, we ignore the legacy MAC in order to concentrate on XG. Thus, without loss of generality, we use the rightmost diagram as the conceptual basis for the development of the architecture.

We now consider a first-level decomposition of the XG MAC and physical layer functionality. XG is mostly a MAC level system, however, some of the key pieces it requires are arguably at the physical layer[10]. One example is *sensing* – the collection, and possible averaging of received signal strengths, perhaps over a wide bandwidth. This requires some consideration of *cross-layer* issues. A goal is to cleanly manage such issues using the XG Transceiver API.

The modular decomposition is given in Figure 33 below. There are three high-level modules: *Opportunity Awareness, Opportunity Allocation,* and *Opportunity Use*. We define their functions briefly below, and elaborate them later.

- *Opportunity Awareness*. This determines the set of available opportunities and associated constraints on their use. This set is dynamic, that is, changes as a function of time. The opportunity availability is determined for a subset of XG nodes, typically in the neighborhood (within a certain radius) of the given node. The opportunity awareness function is a distributed procedure that may include any or all of the following – sensing of spectrum opportunities, identification of usable opportunities and associated constraints, and the dissemination of this information to an appropriate neighborhood.
- *Opportunity Allocation*. This is a distributed procedure that allocates the available opportunities (as determined by the opportunity awareness module) for transmission amongst the XG nodes. The allocation is dynamic, that is, changes with time. The opportunity allocation may be done based on any medium access control approach – CSMA/CA, FDMA, TDMA, CDMA, or a combination thereof. Clearly, the mechanism depends upon which approach is used, but the functionality itself is agnostic to the actual mechanism. However, the mechanisms that can be used depends upon the awareness information available (for instance, if no code opportunity information is available, one cannot exploit code opportunities and allocate them).
- *Opportunity Use*. This refers to the physical layer mechanism that achieves transmission of a set of packets over the set of indicated opportunities. That is, its job is to ensure that a packet is transmitted as quickly as possible subject to constraints supplied to it (e.g., transmit on frequencies *f1-f7*, at power level not to exceed *p*, and a spreading of at least *k* chips/bit). Clearly, a large number of possible opportunity use mechanisms exist, from sequential channel access to morphed waveforms. Again, the module does not dictate how it should be done, merely what is to be done.

[10] One may ask: why do we need to bother about which layer something is? Why not simply treat the entire XG system as one big "box" and think about a decomposition. This is possible – in the same way that it is possible to think of a router without using layers. The fact is that layering allows conceptual clarity and a certain relationship to existing functions which is helpful.

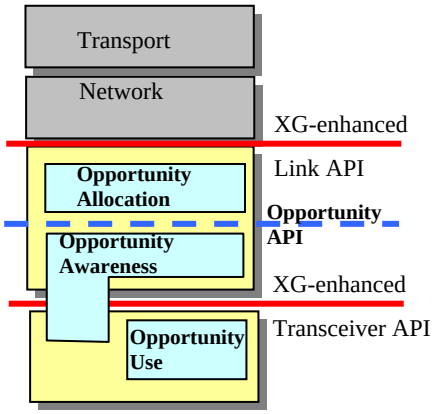


Figure 33: XG Modules in a layering context

Figure 33 also shows an API – the opportunity API – between the allocation and awareness modules. This API helps to cleanly separate two functions – determining opportunities, and using them. This allows independent progressive refinement of each, and a large number of solutions for each within the same framework. Although it is an XG-internal API, it is a significant one because it separates the behaviors that are likely to have regulatory implications from those that will largely be outside of regulatory purview.

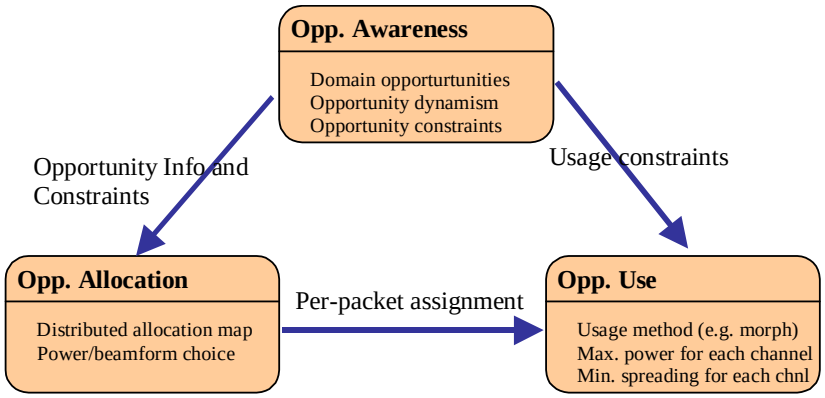


Figure 34: Top-level XG modules

Figure 34 above shows an example of the interaction between modules and an example of information resident in each.

The opportunity awareness module tracks the opportunities in the XG *domain* (refer to section 3.3), and its dynamism, including perhaps information to predict the availability of opportunities. It also tracks the constraints on the opportunities, such as time window, maximum power and other transmitter parameters that need to be used to use that opportunity. The identification of opportunities is controlled by policies.

The opportunity allocation module uses the opportunity information and constraints from the awareness module and creates a dynamic allocation map. The allocation map is essentially a distributed database of frequency, and possibly time slot or code assignments to XG nodes. This module also tracks information such as the power and beamforming to use for that assignment, etc. For a given packet, it then provides the assignment of opportunity to the opportunity use module.

The opportunity use module tracks current preference of usage method (if there are multiple), and maintains a set of lower and upper bounds on transceiver parameters for interference preservation. Such bounds are created using information from the awareness module which can supply usage constraints. Arguably, the allocation module can provide this information too, as it knows (and has to know) the usage constraints. However, the flow of information in the architecture is more streamlined if this information were given directly. Furthermore, this information relates directly to behaviors that are closely associated with interference preservation. By removing this from the allocation module, we allow for the possibility that all of allocation is "non-core", that is outside of regulatory purview. This would be a step in the direction of our having a small set of core behaviors within a boundary and allowing innovation outside of it (see the Vision RFC for more on this).

We emphasize that the information depicted in Figure 34 is only an example. A number of other pieces of information are relevant and will be presented in a more detailed version of the design.

6.1 Opportunity Awareness

We now consider the opportunity awareness module in more detail. This is the key module for XG, and undeniably the most significant from an architectural viewpoint. A natural decomposition of this module is in terms of *sensing, identification* and *dissemination.*

- *Sensing.* This is the process of sampling the channel in order to determine occupancy. We note that there is no fixed definition of when a channel is occupied – it depends upon the receiver (its sensitivity for instance), the sampling window, the average and peak values within the window, thresholds on discriminating noise from signals etc. The criteria for declaring a channel occupied may also change with time. However, the basic notion is to determine if there is a signal, and if so, what the characteristics of the signal are.
- *Identification.* This is the process of determining whether a channel is an *opportunity*. Note that sensing merely tells you the characteristics of the channel. Identification, on the other hand, uses this information to determine whether or not it can be used by XG. If a channel is sensed free, it may or may not be prudent to use it (maybe we are in a deep fade). Similarly, even if a channel is occupied, it may be acceptable to transmit within a power level. Thus, identification contains the algorithms to convert sensed information to be useful to XG.
- *Dissemination.* As mentioned earlier, opportunity awareness needs to include not just the node but also some subset of its k-hop neighbors. This is because allocation mechanisms often do much better with somewhat global knowledge. Dissemination is the process of distributing the information to other nodes so that opportunity awareness to the extent necessary is achieved. The phrase "to the extent necessary" captures a whole slew of possible mechanisms, each interacting with a possible allocation mechanism. This is deliberate, and is an example of the range of innovations possible and the need to support such innovations in the regulatable kernel.

Figure 35 below shows an example of the interactions between the submodules of opportunity awareness. As in Figure 34 the information held by each and the interaction between the submodules is also depicted.

The sensing submodule tracks the signal level characteristics in the channel, perhaps even identifying it as primary or secondary. It might also keep track of the activity statistics, or the most recent history of activity. If the sensitivity threshold can be adjusted, it may store the current value used. The sensing submodule provides the identification submodule channel activity information to help determine whether or not it is an opportunity. At some logical, goal-specific level, the information that flows from

sensing to identification is the set of *possible* opportunities. The identification submodule determines which of these are *real* opportunities, and how they can be used. Accordingly, the marking of a channel as an opportunity is tracked by the identification submodule, as is the expected lifetime of the opportunity and constraints on its use (such as maximum power).

The "real" opportunities are then passed to the dissemination submodule that is responsible for collecting the local opportunity information at various nodes. Accordingly, it maintains the local opportunity information as well as opportunity information for the relevant sub-domain in the XG node's neighborhood, and related constraints. This is the data that is provided to the opportunity allocation module. Dissemination takes time and network resources. Therefore, the architecture supports multiple maps of the relevant sub-domain, for instance, an accurate (up-to-date) view of coarse granularity and an approximate (out-of-date) map of fine granularity. Different allocation schemes may want to use these different levels of information.

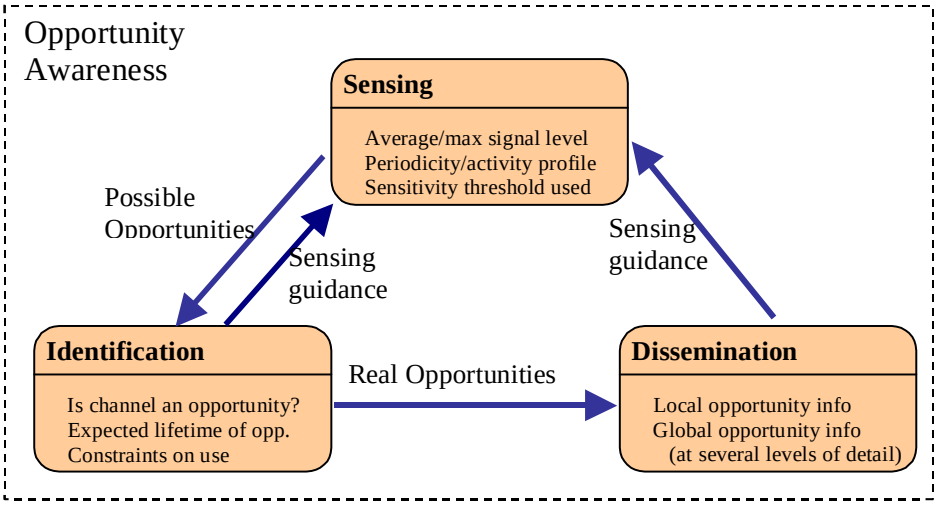


Figure 35: Submodules of Opportunity Awareness

Finally, the identification and dissemination processes may provide information to the sensing module to enable better or more efficient sensing. For instance, if it is determined that a channel is likely to be an opportunity for the next several hours, or if it is determined that a channel should be left alone for a period of time, the sensing module can skip that channel and save time.

We emphasize that the information depicted in Figure 35 is only an example. A number of other pieces of information are relevant and will be presented in a more detailed version of the design.

7 XG Architecture: Near-Term Usage Examples

Recent measurements have shown that a typical geographical region has wide swathes of spectrum where there are no users at all. Thus, even without sophisticated predictor-corrector or dynamic management techniques, one can dramatically improve system capacity and enable rapid entry into an area without apriori frequency assignment.

The "near term usage examples" in this section is aimed at plucking such "low hanging fruit". Another way of looking at the goal of this architecture is: what is the simplest set of techniques/protocols for opportunistic use of spectrum? In particular, the goal here is not the *optimum* use of resources, but the *easiest* way to reasonably utilize gaps in spectral occupancy. XG must be able to function along with existing technologies without requiring any modification of them. XG functionality must be inserted as

unobtrusively as possible into current architectures. The near-term usage examples are based on the middle diagram in Figure 32. We will refer to this as the XG *near term* architecture and reiterate it below in Figure 36.

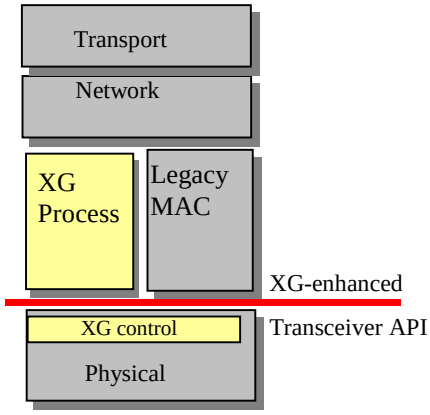


Figure 36: XG *near term* architecture

As envisaged in the Vision RFC, the near-term XG architecture allows the co-existence of the XG process with a legacy MAC. Specifically, we consider a CSMA/CA (as in 802.11 a/b/g) MAC and give examples of how the XG process might inject a modicum of dynamic spectrum management using very simple strategies.

In this framework, the XG control detects Legacy MAC (L-MAC) frames that have XG implications. These generate requests through the XG-enhanced Transceiver API to the XG process that triggers certain behaviors in order to satisfy the L-MAC requests. Such behaviors may involve the encapsulation/decapsulation of the L-MAC packets (transparently to L-MAC), or simply the selection of appropriate XG physical layer services for the transmission/reception of the L-MAC packets. The XG process may also exchange opportunity information with peer XG processes to do intelligent, coordinated sensing. It might also use it for disseminating opportunity information for resource allocation purposes.

We illustrate the concept of operations with this architecture using a few examples, identified as *XG-less fallback, Zero control, Frequency selection,* and *Frequency negotiation.* In each case, we identify the peer interactions and packet modifications that might be necessary. For these examples, we assume a 802.11-like CSMA/CA protocol. However, we note that this is only for illustration and the architecture by no means restricts the kind of legacy protocol. Also, we emphasize that these are only examples – the fact that embellishments of 802.11 are suggested does not alter our goal of keeping the architecture, and this RFC at a general level.

XG-less Fallback

When used with other legacy radios, the architecture defaults to a “no operation” with respect to XG functionality as follows. All L-MAC protocol data units (PDUs) are simply sent out without any processing by the XG control. This could happen based on configuration or set as the default case, with XG functionality being invoked only upon discovering the presence of one or more XG-capable nodes.

Zero Control

In this example concept of operations, we assume that the transceiver has wideband tuning capability on receive and frequency agility on transmit. The XG process senses a contiguous set *S* of channels that are completely unoccupied in that geographical area. When receiving, the node listens on all of the channels

in the contiguous set *S* and when detecting a packet on one of the channels, it tunes to that channel. When transmitting, XG picks one of the channels at random and sends the packet. The state for this behavior is placed in the XG control by the XG process, so that the packet does not have to travel back up to the MAC layer. The RTS/CTS/DATA/ACK packets are sent as-is, except that a selection of channel from among those unoccupied is made via the XG Transceiver API.

This could allow multiple parallel communications on different channels to occur within a geographical area in the simplest possible manner. We note that the legacy MAC, for example 802.11, can be completely oblivious of the XG functionality.

Several behaviors are possible upon collision of, say the RTS, due to two nodes picking the same channel. The XG process could simply not take any action, but rely on the L-MAC retransmissions (which might result in a new unoccupied channel being picked by the random process). Or the XG process could, through the XG control, itself attempt retransmissions, or proactively send multiple RTSs on different random channels. We note that some of these behaviors may have interactions with a legacy MAC. For instance, if XG attempts its own retransmissions, an XG-unaware L-MAC would timeout on the RTS.

Frequency Selection

This assumes the existence of an a priori dedicated control channel that is known to all nodes. The RTS/CTS are sent on the control channel and used to select a channel for the DATA/ACK. The XG process identifies the set of possible channels by sensing. There is no need for the channels to be contiguous. There is also no need for the transceiver to be wideband tunable, but it should be frequency agile. When in idle, the transceiver is tuned to the control channel.

In the context of the architecture, the operation is as follows. The XG control encapsulates the RTS into a new packet, say, X-RTS. The X-RTS contains the suggested channel number *c* for the DATA/ACK communications, chosen randomly from among those available. The peer XG control of the receiver decapsulates the X-RTS and sends it to its L-MAC. It also notes whether the channel *c* is usable or not. If it is, then the corresponding CTS is encapsulated into an X-CTS, conveying that this channel is fine. For the DATA and ACK, no encapsulation is necessary, but the physical layer is directed to select the channel *c* (similar to zero control). Peer XG control modules of nodes that receive the X-RTS or X-CTS simply decapsulate and pass it to their L-MACs that perform the usual NAV operations. Additionally, they also note which channels have been chosen or in the process of being chosen, so that they can avoid those channels for selection to put into the X-RTS or X-CTS.

Once again, we note that the architecture allows for the above concept of operations to happen without the legacy MAC being XG-aware.

Multiple Frequency Negotiation

This is similar in spirit to the frequency selection, except that multiple frequencies are chosen, and there is scope for the receiver to pick a subset of the frequencies it is offered. As in frequency selection, we assume the existence of an a priori dedicated control channel that is known to all nodes, transceiver should be frequency agile, and we assume that sensing is done independently by the XG process. The identified frequencies may be non-contiguous. The frequencies negotiated for communication may be contiguous or non-contiguous – if the latter, then it is assumed that the physical layer has the capability to send a packet over non-contiguous frequencies.

In the context of the architecture, the operation is as follows. The XG layer encapsulates the RTS into X-RTS, listing a number of possible channels that can be used. The X-CTS is returned with a subset of these channels, and the DATA and ACK use one or more (if non-contiguous frequencies can be used) of these channels. As before, nodes other than the transmitter and receiver keep track of the chosen channels using the overheard X-CTS.

Once again, we note that the architecture allows for the above concept of operations to happen without the legacy MAC being XG-aware.

This transparency, however, may lead to sub-optimal performance. Consider for instance the case when a morphed waveform is used to transmit the DATA over multiple frequencies, thereby shortening its transmission time. Ideally, this should allow other nodes start their pending transmissions sooner than if the DATA had been sent on a single channel. However, the fact that the L-MAC is XG-unaware means that the NAV period will not be adjusted, and hence the nodes will continue to be backed off for a time equal to if a single channel had been used.

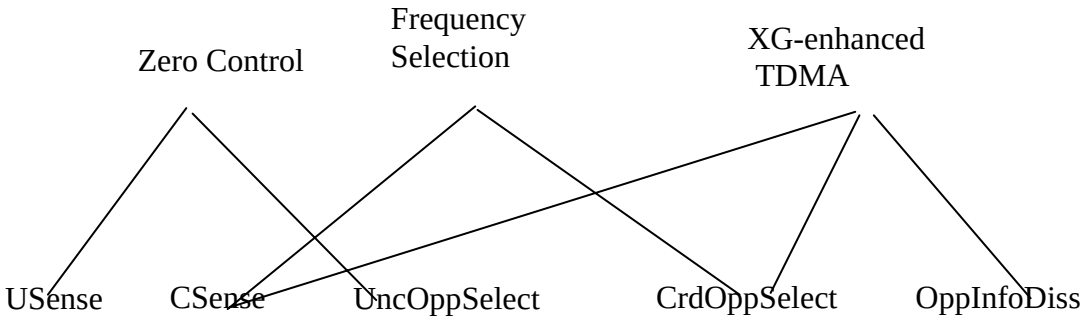
8 Abstract Behaviors

A key goal of the RFC process is to specify a set of abstract behaviors for XG. The first step toward that, of course, is to decide: *what are the abstract behaviors that should be specified?* This section "gets the ball rolling" by identifying a set of possible behaviors. Recall that the specifications of the behaviors themselves are part of the protocol RFCs.

Following our modular breakup, we can identify five top-level behaviors: sensing, identification, dissemination, allocation and use of opportunities. Each of these behaviors may be represented as an abstract class with resident data and methods for access. At a second (lower) level of abstraction, we can identify behaviors that correspond to different ways of achieving the desired top-level behavior. These include:

- *Uncoordinated Sensing*. The sensing here is completely local to the node, and the opportunity is identified based solely on spectrum occupancy as seen by this node.
- *Coordinated Sensing*. Control messages are exchanged between nodes in order to implement a "quorum" based decision on opportunities.
- *Uncoordinated Opportunity Selection*. The selection of an opportunity for DATA transmission is done without consultation with the peer(s) involved in the communication. For example, Zero Control employs this.
- *Coordinated Opportunity Selection*. The selection of an opportunity for DATA transmission is done based on one or more handshakes that might involve XG control messages. For example, the X-RTS and X-CTS control messages are used in the Frequency Selection approach.
- *Opportunity Information Dissemination (OID)*. Dynamic opportunity information is exchanged between nodes in a neighborhood so as to help coordinated opportunity selection. This might involve control messages that convey such information over multiple hops so as to make more efficient allocations.

As an example, the mechanisms described in section 7 can be seen as combinations of some of the above behaviors. This is illustrated in the figure below.



Each mechanism is appropriate for a different set of hardware and assumptions: e.g., zero control when no control channel, but wideband tuning on listen; multiple frequency negotiation when control channel and ability to spread a packet over multiple non-contiguous frequencies etc.

The total number of possible mechanisms is potentially large, but the set of core behaviors could be much smaller. By itself, a core behavior may not provide all the XG functionality required to dynamically use spectrum. However, if regulatory bodies approve each behavior, then the entire mechanism's behavior is approved. For instance, if one can show that all of the core behaviors are *interference preserving* – that is, the introduction of a signal will not degrade the performance of any then operating system by more than a set threshold – then the mechanisms constructed out of these behaviors will also be interference preserving.

Another advantage of this approach is that given the core behaviors, perhaps the mechanism best suited for the assumptions and hardware can be constructed on the fly.

We note that these are just initial ideas for how to identify the right set of behaviors to specify, and expect them to change as thinking evolves. Abstract behaviors will be specified in more detail in the protocol RFCs.

Acknowledgments

This document was prepared by the Internetwork Research Department, BBN Technologies, with input from DARPA/ATO, AFRL, DSO, MITRE, Raytheon Corp., and Shared Spectrum Corp.

Comments

Comments on this RFC should be emailed to Ram Ramanathan at ramanath@bbn.com, along with the commenter's name and organization.

Appendix C
XG Abstract Behaviors
Request for Comments

Version 0.6.4
December 30, 2004

1 Introduction

1.1 Purpose and Scope

This document describes the abstract behaviors of XG systems.

This document is a Request for Comments (RFC). Accordingly, an important purpose of this document is to obtain feedback from the community at large, and to refine the ideas described here based on that feedback. The development of the XG abstract behaviors is an evolutionary process, and this document reflects a snapshot in that evolution.

This RFC is one of a series of documents intended to describe the basic elements of an XG system. We recommend that this document be read after the *XG Vision RFC* [XGV]. The *XG Vision RFC* lays out the motivation for XG and its scope, presents the key concepts underlying XG, and describes an approach for defining XG. This RFC also refers to the *XG Architectural Framework RFC* [XGAF] and the *XG Policy Language Framework RFC* [XGPLF]. The *XG Architectural Framework RFC* [XGAF] presents the architecture, system components, and a high level concept of operations for XG communications. The *XG Policy Language Framework RFC* presents a language framework to express, interpret, and enforce spectrum policies for XG systems.

1.2 Document Organization

The rest of this document is structured as follows. In this section, we provide a motivation for, and an overview of, abstract behaviors of XG radio systems. We also present a notional XG radio system model, and describe the notation uses.

In Section 2, we present an UML model of an abstract XG radio system. We present three kinds of abstractions – *Interfaces*, *Behaviors*, and *Information Objects* – in terms of which an abstract system can be described. We provide a high level object-oriented description of the abstract XG radio system in terms of these three kinds of abstractions.

The next three sections describe each of the three kinds of abstractions in detail, namely Interfaces in Section 3, Behaviors in Section 4, and Information Objects in Section 5.

In Section 6, we describe a reference radio design based on the abstractions described in this document. We conclude with a summary and a discussion of future work required in this area in Section 7.

1.3 Motivation

The goal of the DARPA neXt Generation communications program (XG) is to enable the use of wireless communications platforms that opportunistically share the wireless spectrum. Opportunistic sharing requires radios to be able to adapt their behavior to the current wireless environment, including respecting local rules regarding spectrum use and co-existing with legacy emitters.

XG systems are expected to use highly adaptive and agile transceivers. This agility enables new and efficient use of spectrum, through a wide range of design and implementation choices. At the same time, this agility poses a major challenge to regulators charged with protecting other spectrum users from interference, who must test and certify these systems for policy compliance, and system implementers, who wish to ensure that their XG device is well behaved. Additionally, spectrum assignees that may wish to rent unused capacity will likely want assurances that devices follow the incumbent assignee's restrictions on how unused capacity is accessed and shared.

In order to facilitate the verification, validation, and accreditation process for a wide range of diverse XG systems, it is desirable to identify and formally specify the minimal set of behaviors that a system much implement to be able to safely share or use available spectrum and conform to the requirements of regulators and spectrum assignees. In short, if a device meets this minimal set of requirements, its opportunistic use of the spectrum will be "safe" from the perspective of regulators and incumbent spectrum assignees.

The challenge in specifying a minimal set of behaviors is, of course, that the range of envisioned XG system is large. An XG system could be a simple transceiver capable of employing IEEE 802.11 signaling on any one of sixteen different frequencies, or an XG system could be a fully software-programmable radio capable of transmitting at any frequency in the range from 3KHz to 300 GHz using any one of a hundred encoding techniques. The challenge is how to specify behaviors, without requiring extraneous functionality for the simple system and without under-specifying the behavior of the fully programmable system. At the same time, the set of behaviors should be all of those necessary and sufficient for regulators (and incumbent spectrum assignees) to deem a radio safe and compliant, so that a new set of guidelines is not required for certifying each new XG implementation.

The solution to this challenge is to specify abstractions – *behaviors, interfaces and information objects* – that allow considerable flexibility in how the abstractions are implemented.

An abstract behavior is a specification of the behavior of an XG system. Abstract behaviors interact with each other and with the system via abstract interfaces.

An abstract interface is an interface that specifies a set of functions (but not their implementations), and an abstract information object specifies the information that passes across the functional interface.

A useful way to think of an abstract interface is as a checklist: for each function in the abstract interface, an XG system developer needs to show their system implements that function. The function may be implemented using modularity very different from the specification in this document. Indeed, in some systems, there will be functions that are implicit. For instance, consider an 802.11 radio with sixteen channels; if the abstract interface has functions to select signaling and encoding schemes, these functions are implicitly implemented, in the sense that the radio can only make one choice (802.11).

In this document we specify a set of abstract behaviors that XG-compliant systems must implement; we include these behaviors in the ***accreditable kernel*** and argue that it is those behaviors that are necessary and sufficient to satisfy policy requirements and demands of spectrum assignees without requiring the development of a new set of guidelines for certifying each new XG implementation.

Related to the notion of an accreditable kernel is the notion of ***traceability*****.** An XG system implementation, therefore, must support a one-to-one mapping between the external physical behavior (e.g. emission profile) of the system and the instances of particular abstract behaviors that the system implements. Linking external behavior (which is fundamentally what spectrum regulators and incumbent assignees care about) with internal abstract behavior provides both a standard for minimalism (if it doesn't affect external behavior, it isn't a function or behavior in the minimal set) and suggests an approach to auditing radio behavior, to ensure the radio is well behaved.

Following this rationale, we have developed a set of guidelines to help determine whether a given abstract behavior must be included in the accreditable kernel or not. In order to be included, the behavior must satisfy at least one of the following:

- It is required to support opportunistic sharing of spectrum
- It is required to support policy-defined operation
- It provides traceability from policy through behavior to emissions

The abstract behaviors are only intended to provide guidelines for the high-level system design of the individual XG systems. Although the abstract behaviors will cover the important attributes that characterize XG systems, they are not intended to be detailed design- or implementation-level specifications of particular XG systems.

1.4 Abstract Behaviors Overview

The fundamental requirement of XG systems is to sense and use underutilized (or unused) spectrum as authorized by applicable policy. At the highest level this requirement leads to the following three classes of abstract behaviors for XG systems as identified by the XG Vision RFC [XGV]:

Awareness: XG systems must determine the set of available opportunities to utilize portions of the spectrum and the associated constraints on exercising those opportunities. This set is dynamic, that is, it changes as a function of time. An available opportunity is determined for a subset of XG nodes, typically in the neighborhood (within a certain radius) of the given node.

The awareness function can be either a local or a distributed procedure, and may include any or all of the following: – sensing of spectrum opportunities, identification of usable opportunities and associated constraints, and the dissemination of this information to an appropriate neighborhood.

Awareness behavior is decomposed into two parts: internal behavior (spectrum awareness) and external facing behavior (awareness dissemination).

Allocation: XG systems must allocate the available opportunities for transmission amongst the XG nodes, either explicitly through communication among the nodes, or implicitly by following local rules intended to ensure consistent (or non-conflicting) behavior. The opportunity allocation may be done based on any medium access control approach: CSMA/CA, FDMA, TDMA, CDMA, or a combination thereof.

Clearly, the mechanism for allocation depends upon which approach is used, but the functionality itself, the fact that an allocation must be made, is agnostic to the mechanism used to realize the allocation. At the same time, it is important to observe that the mechanisms that can be used depend upon the awareness information available (for instance, if no code opportunity information is available, one cannot allocate code opportunities).

Allocation behavior is concerned with choosing a specific opportunity among the available opportunities. Because the range of opportunities will change over time, as new transmission methods and new sensing technology is developed, it is desirable to seek to keep allocation out of the accreditable kernel. The fact that opportunities will change also makes allocation behavior a poor choice for policy regulation. So, this description takes the view that as long as the opportunities identified conform to policy, and the opportunity allocated by this behavior is used in a manner that conforms to policy, this behavior can be kept outside the accreditable kernel. Quite simply, this behavior offers the most room for innovation (e.g. novel spectrum arbitration and brokerage mechanisms) and potential for optimization (e.g. scheduling algorithms). Therefore we do not seek to itemize this behavior; and instead only identify the need for an interface through which opportunity allocations may be obtained from system-specific procedures.

Use: *Use* refers to the physical layer mechanism that achieves transmission of a set of packets over the set of indicated opportunities. That is, its job is to ensure that a packet is transmitted as quickly as possible subject to constraints supplied to it (e.g., transmit on frequencies *f1-f7*, at power level not to exceed *p*, and a spreading of at least *k* chips/bit). Clearly, a large number of possible opportunity uses

mechanisms exist, from sequential channel access to morphed waveforms. Again, the module does not dictate how it should be done, merely what is to be done.

Use behavior is decomposed into an internal behavior (Usage Accounting) and a protocol behavior (Use Coordination).

1.5 Tools and Notation

We describe the XG abstract behaviors through an object-oriented specification. We make use of the popular Unified Modeling Language (UML) notation for a visual representation of the concepts (see [UML]).

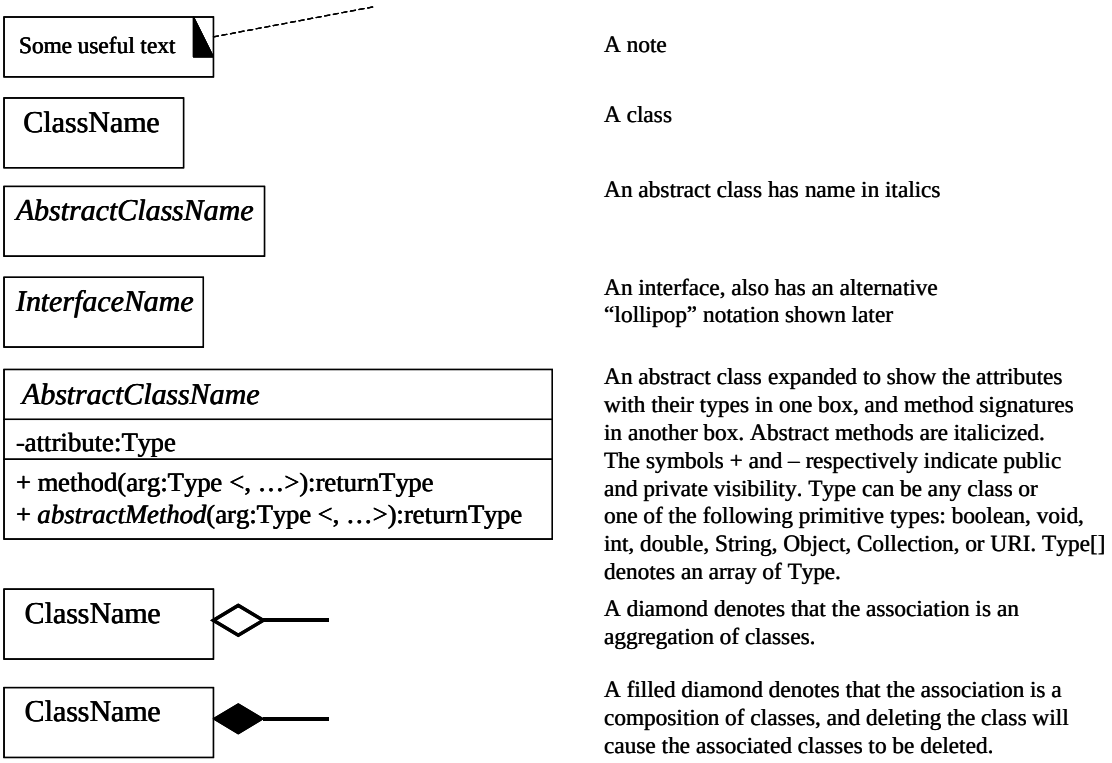
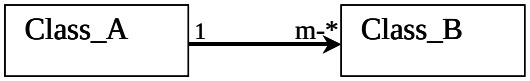
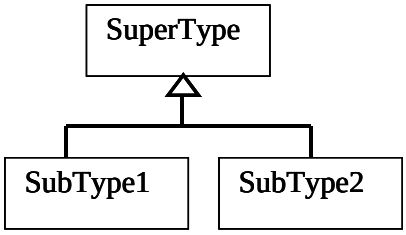


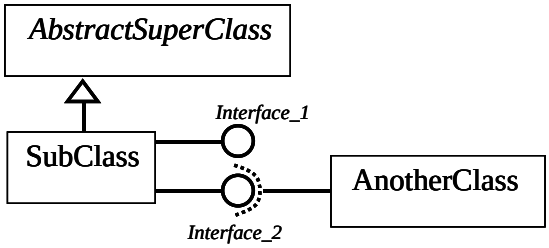
Figure 37: Basic UML notation used in the RFC



A class diagram showing an association between two classes using a solid line. A dotted line would indicate a dependency. The arrowhead indicates navigability of the association. The multiplicity (range) of the association is shown on the line using m (m-n), which can be the wildcard *.

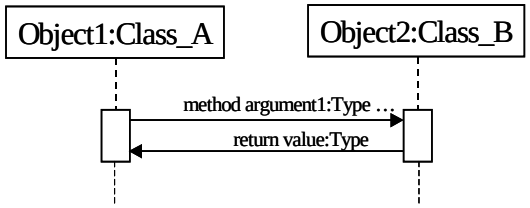


A class diagram showing the generalization of subtypes by a supertype. The subtypes inherit and extend the attributes and methods of the supertype.



A class diagram showing an abstract super class, and a concrete sub class that implements two Interfaces, shown in lollipop notation. Another class that uses the interface is also illustrated.

Figure 38: UML class diagram notation used in this RFC



A sequence diagram showing a method being invoked by Object1 on Object2 with arguments of specified type and the return being shown with arrows. Method calls can be nested and multiple objects can be involved in a sequence diagram. The sequence of method calls and returns follows their order in the diagram from top to bottom.

Figure 39: UML sequence diagram notation used in this RFC

1.6 Notional XG Radio System model

We present a general high-level block diagram of a notional XG radio system for the purpose of setting a context for this RFC. In Figure 40 we illustrate this notional model showing the logical location of the XG abstract behaviors with respect to the key subsystems.

The XG radio system includes sensors to acquire spectral awareness, agile transceivers, and logical link control/medium access control (LLC/MAC) layers. The XG abstract behaviors are implemented within an accreditable kernel, which provides access to system capabilities, configuration, and state. The accreditable kernel interacts with a policy conformance reasoner subsystem to ensure compliance to regulatory policy. A system strategy reasoner provides opportunity allocation function. XG protocols to acquire and disseminate awareness of spectrum and for coordination of spectrum use make use of virtual coordination channels that may be implemented at any layer.
We will use this model throughout this RFC, and we will elaborate on the subsystems and interfaces in

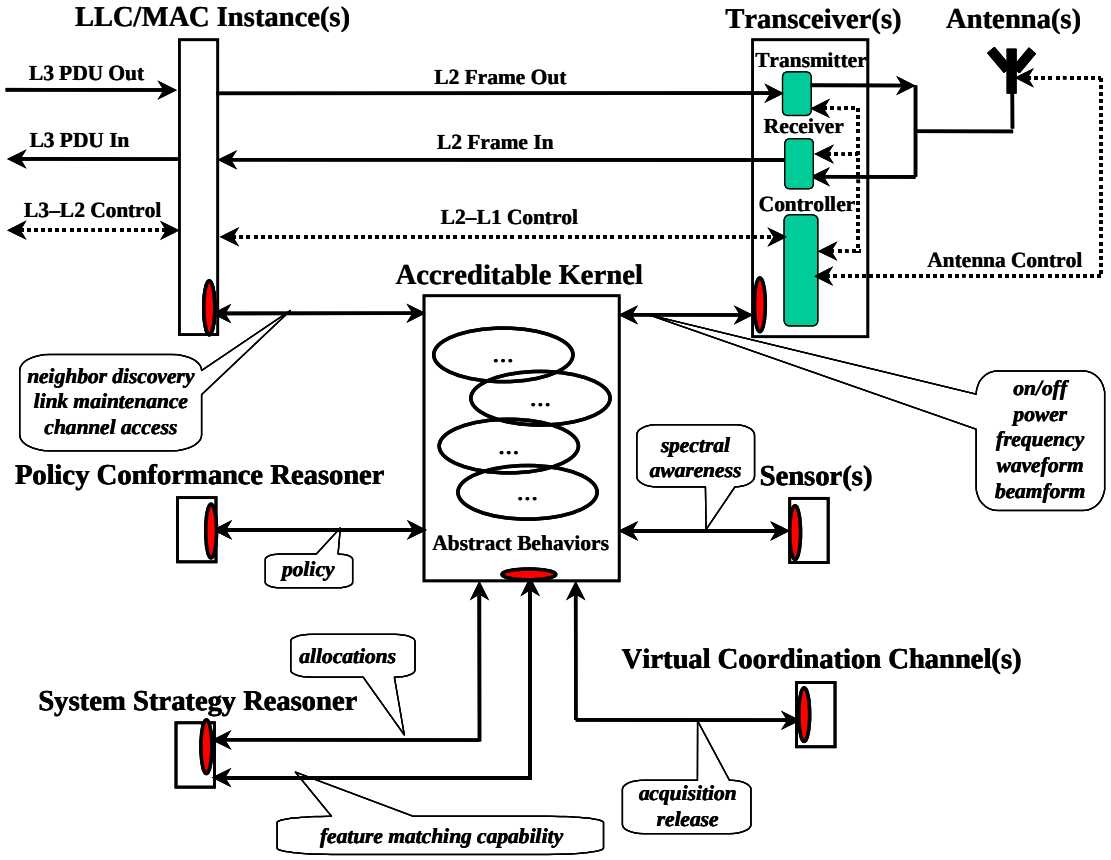


Figure 40: A Notional XG Radio System Model

Section 4.1 Adherence to this notional model itself is ***not*** required. However, any XG system must demonstrate that it implements the abstract behaviors and interfaces surrounding the accreditable kernel, as described below.

2 An Abstract Model of the XG Radio System

In this section, we present an abstract model of an XG radio system (radio plus software on it and its interactions with the outside world). That may appear to be at odds with our goal in this document of defining a minimal set of essential behaviors, our *accreditable kernel*. The abstract behaviors that reside within the accreditable kernel, however, interact (through XG Interfaces) with other portions of the XG radio system. So in defining the accreditable kernel, we also need to present how it interacts with the portions of the radio outside the accreditable kernel. That's the focus of this section.

Since our goal is to define the accreditable kernel, and leave the details of the rest of the radio completely open to innovation (and indeed, to keep the accreditable kernel open to innovation too), this abstract model is a careful combination of concreteness and vagueness. In general, we seek to specifically describe what functionalities have to be present, yet we do not specify the details of how those functionalities are realized.

In this section, we define a UML class for a radio, entitled XgRadioSystem. This class is illustrated in Figure 41 and is composed of all the components that an XG radio might need. Many components are optional, but some components must be present, and some components may be present one or more times.

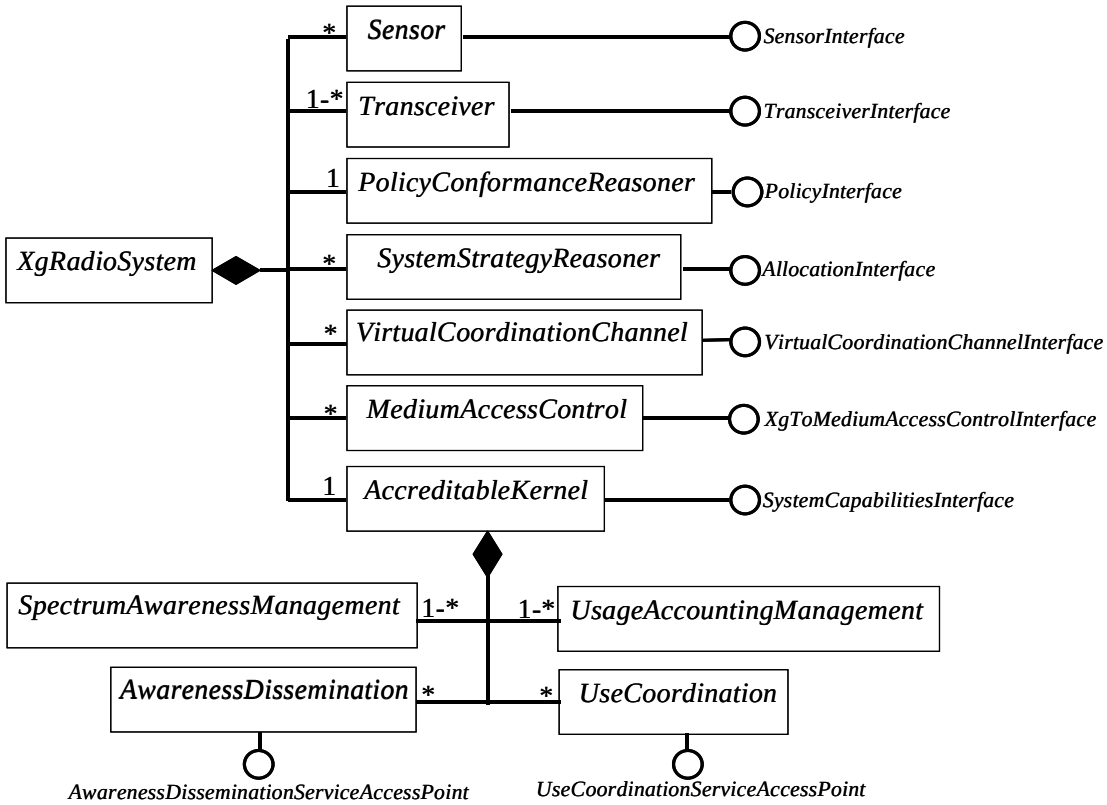


Figure 41: XG Radio System Abstract Model

The purpose of this representation is to illustrate and represent the basic hardware and primitives of a host radio system that enables opportunistic use of spectrum (e.g., RF front end, DSP hardware, system

software including the OS, middleware, and libraries, and primitives for networking protocols, waveform agility, and beam forming). This configuration of components is only one of many ways that one might choose to modularize the functions of an XG radio.

In general, each component of the XgRadioSystem class has a related interface. (It is important to keep in mind, however, that other classes may also implement a particular interface. The mapping shown in Figure 41 simply indicates that the class, if present, implements the associated interfaces). The accreditable kernel interacts with the various components through these interfaces, using information objects. So in this section we describe what each component does, and define a generic class, XG Interface, from which each component's interface will be derived. In anticipation of defining the behaviors of the accreditable kernel, we define a generic class of behaviors, XG Behavior. Finally, we define three generic classes of XG Information Objects from which all information objects used in the interfaces will be derived.

2.1 Subsystems of the XG Radio System

The various components of the XgRadioSystem class serve different purposes. We describe each component and its function here.

2.1.1 AccreditableKernel (required; one instance)

The AccreditableKernel class has the following key functions:

(iv) Implement the included XG abstract behaviors and provide access to them;

(v) Manage access to the state, configuration, and capabilities of the radio; and

(vi) Manage access to primitives implemented within the radio that can be governed by policy.

Every instance of the XgRadioSystem must have an instance of the AccreditableKernel.

The AccreditableKernel class implements an interface (the SystemCapabilitiesInterface) through which it provides access to primitives deemed necessary to ensure that control of the radio's visible behavior is located in the AccreditableKernel and that the behavior is controlled in a fashion consistent with XG policies. These primitives include Parameters and Processes described within the XG Policy Language Framework (see [XGPLF]).

The accreditable kernel also has an association with every other subsystem in the radio for each primitive implemented in that subsystem (these associations are not shown in the diagram above).

2.1.2 Sensor (optional; may have more than one)

The Sensor class provides situational information to the radio about the spectral environment at a given location and time. The sensor outputs need not be limited to the RF spectral environment; the provided information could include a variety of other values such as geo-location, temperature, and proximity to specific targets that could be used as parameters for system policy.

Sensors are optional because radios may learn through other means, such as configuration or through dissemination protocols, what spectrum use local operating rules authorize. Alternatively, the sensor functionality may be implemented within the transceiver subsystem. There may be more than one sensor because a radio may sense different information using different devices.

The Sensor class implements the interface called SensorInterface.

2.1.3 Transceiver (required; at least one)

The Transceiver class provides radio communications capabilities, including the RF front-end and the baseband signal processing functions. In general, we expect these communications capabilities to be agile and that the interface to the Transceiver will reflect this agility.

Instances of XgRadioSystem contain at least one Transceiver, by which we mean it must have at least a receiver or a transmitter capability. (Without a transceiver, we don't have a radio!).

The Transceiver implements the interface called TransceiverInterface. In cases where the transceiver is also a sensor, it may make sense for the transceiver to implement SensorInterface as well.

2.1.4 PolicyConformanceReasoner (required; one instance)

The PolicyConformanceReasoner class determines whether proposed spectrum use is consistent with accredited policy (e.g. approved by a relevant regulatory authority and incumbent spectrum assignee),

knowledge of the local spectrum (e.g. information from the Sensor class), and other background knowledge. Note that the PolicyConformanceReasoner primarily determines if proposed use is acceptable – the task of developing proposed usages is left to other subsystems, for example, the SystemStrategyReasoner. The PolicyConformanceReasoner may, however, perform additional policy functions. It can support queries on policy to extract authorized usages for a given situation, support filtering policy information to obtain a subset of policies that apply to a current situation, generate machine proofs that a given usage conforms to policy, and manage the loading and revocation of policy sets.

Instances of XgRadioSystem must contain a PolicyConformanceReasoner, as the PolicyConformanceReasoner implements the functions that determine if the XG radio is in compliance with any policy restrictions and the abstract kernel depends on conformance reasoner to accredit usages

The PolicyConformanceReasoner class implements the interface called PolicyInterface.

2.1.5 SystemStrategyReasoner (optional; may have more than one)

The SystemStrategyReasoner class determines the system's strategy for opportunistic spectrum sharing given the constraints of sensor information, regulatory and system policy constraints. This reasoner is aware of system-specific optimizations and tradeoffs and has control over the radio platform.

In many ways, the SystemStrategyReasoner is the complement of the PolicyConformanceReasoner. The conformance reasoner determines whether a particular type of spectrum use is authorized by policy in the current environment; the SystemStrategyReasoner determines what opportunities to use spectrum exist in the current environment that are suitable for the XG radio. It is important to note that the strategy reasoner is not the only place in an XG radio that can identify or allocate opportunities: opportunities can be found from other nodes through an awareness dissemination protocol or this function can be incorporated within a medium access control protocol.

The SystemStrategyReasoner class, therefore, is one of several classes that can perform the opportunity allocation function, and so it implements the interface called the AllocationInterface. This class can access the state, capabilities, and configuration information through the SystemCapabilitiesInterface.

2.1.6 VirtualCoordinationChannel (optional; may have more than one)

In many situations, XG radios need to communicate with neighboring XG radios to coordinate awareness of the spectrum (e.g. are a set of frequencies unused at both sender and potential receivers?) and coordinate use (e.g., jointly determine which frequency bands will be used). Coordination channels can be pre-configured, discovered, or created when needed. Furthermore, the channel may be implemented using waveform-level signaling, as a specialized MAC layer, or even at the application layer using higher layer protocols using out-of-band network access.

The VirtualCoordinationChannel class represents the logical communication channel used for this purpose.

The VirtualCoordinationChannel class implements the interface called VirtualCoordinationChannelInterface.

2.1.7 MediumAccessControl (optional; may have more than one)

The MediumAccessControl class includes higher layer functionality such as neighbor discovery, topology management, and link scheduling and sends/receives Layer 2 protocol data units to the transceiver. In an opportunistic spectrum-sharing environment, the higher layers need to interact with

the XG radio. For example, links must be scheduled such that communicating peers select and use common opportunities that are deconflicted from opportunities used by other nodes.

The MediumAccessControl class implements the interface called XgToMediumAccessControlInterface.

2.1.8 Comments on the XgRadioSystem class

The subsystems contained within the XgRadioSystem class are meant only to provide an overall system context. They are not intended to suggest the adoption of a particular system design. In order to allow significant room for innovation and diversity of design, in the rest of the document we will focus on extensible specifications of the XG Behaviors, the XG Interfaces, and the XG Information Objects that XG systems must inherit, extend, and implement concretely.

Particular XG radio system instantiations will extend and implement concrete instances of these abstractions. XG system designs, however, need not adhere to the rest of this abstract model or the system decomposition presented here. They can be based on alternative system decompositions, and they can also integrate multiple functions into fewer subsystems. For example, a particular design might integrate sensor functionality within transceivers. It is important however that each XG system design maintains a clearly identifiable *accreditation boundary* such that spectrum policy concerns are within the boundary, and to the extent possible, other concerns are left outside the boundary.

2.2 XG Interfaces

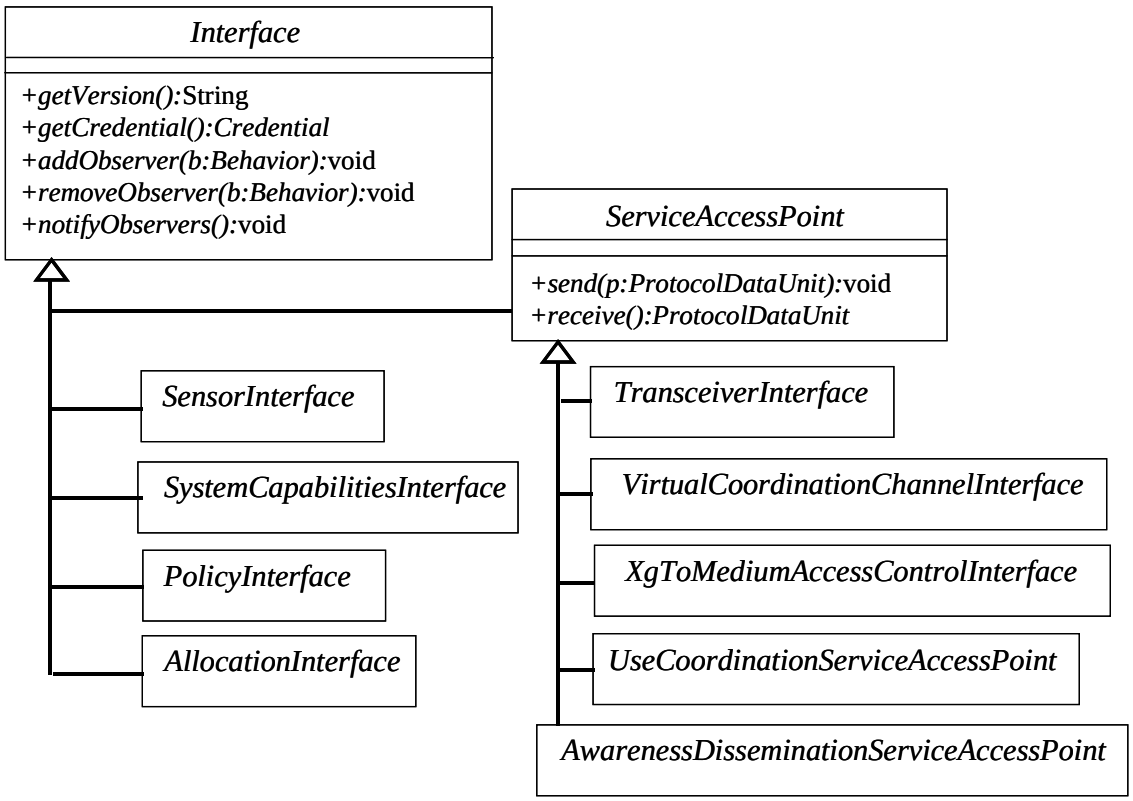


Figure 42: XG Interfaces

In this subsection, we sketch the functions of generic Interface class and briefly mention the purpose of each of the major interfaces in XgRadioSystem. A full description of each interface is provided in Section 3.

As illustrated in Figure 42, all XG interfaces extend the Interface abstract class, or its subclass ServiceAccessPoint. In addition to the seven interfaces associated with the subsystems described in Section 4.1.1, we define two additional interfaces that are associated with the protocol behaviors included in the AccreditableKernel.

The Interface class provides five methods:

- *getVersion*, a method used to learn the version of the Interface; and
- *getCredential*, a method used to get credentials of the class that implements the interface, for example, digital certificates that assert that the implementation has been accredited. *getCredential* allows trust establishment mechanisms to be established to work across XG interfaces. This enables various components of the radio to be accredited and enhanced separately, and the use of particular combinations to be governed by policy if needed.
- a set of three methods that behaviors use to register and unregister at an interface and to be notified of events. Behaviors and Interfaces together implement an Observer–Subject design pattern [DP]. A behavior can register or unregister itself at an interface by calling the *addObserver* and *removeObserver* methods respectively. The *notifyObservers* method is called, which in turn calls the *update* methods on all instances of Behavior that are registered with the

interface. For example, the SpectrumAwarenessManagement behavior can register with the SensorInterface and the AwarenessDisseminationServiceAccessPoint in order to know when new sensed awareness information or protocol based awareness information is received.

The ServiceAccessPoint class extends the Interface class with two additional methods:

- *send*, a method to send a protocol data unit. Note that the definition of protocol data unit (PDU) is somewhat broader in this document than is traditional – in this context, a PDU is protocol data plus an a set of associated attributes which often includes information about what channel the PDU is to be sent on
- *receive*, a method to receive a protocol data unit, typically invoked by an observer in response to a notification

XG abstract behaviors interact with each other and with the system through the following extensible interface set:

1. **SensorInterface:** The interface through which sensed awareness (of the operational environment, i.e. information such as location and spectrum) is accessed and sensing behavior is controlled.
2. **TransceiverInterface:** The interface through which the agile XG transceiver (i.e. parameters such as transmit power, frequency, waveform, and beamform) is controlled and emission constraints are conveyed.
3. **SystemCapabilitiesInterface:** The interface through which the capabilities, the current configuration, and state of the XG system can be accessed.
4. **PolicyInterface:** The interface through which opportunity instances are validated against applicable regulatory, incumbent spectrum assignee, and system policy. This interface may additionally provide access to policy information and policy management directives.
5. **AllocationInterface:** The interface through which a specific opportunity is allocated for use from all available opportunities.
6. **VirtualCoordinationChannelInterface:** The interface through which virtual control channels (which carry XG protocol data) are managed and accessed.
7. **XgToMediumAccessControlInterface:** The interface through which the MAC layer can interact with the XG abstract behaviors (for example, to support link setup and maintenance, contention management, and framing).
8. **AwarenessDisseminationServiceAccessPoint:** The interface through which the AwarenessDissemination protocol behavior can be accessed. The information accessed through this interface is processed spectrum awareness that is acquired from or is to be disseminated to other nodes by the protocol (rather than the individual protocol data units exchanged by a protocol implementing this behavior over virtual coordination channels)
9. **UseCoordinationServiceAccessPoint:** The interface through which the UseCoordination protocol behavior can be accessed. The information accessed through this interface include higher level protocol directives resulting in acquisition and release of opportunities (rather than the individual protocol data units exchanged by a protocol implementing this behavior over virtual coordination channels)

XG systems that implement these interfaces are anticipated to inherit and extend these interfaces and the related classes in order to support real-time operation, access control, error or exception handling, enhanced features and system-specific optimizations.

2.3 XG Behaviors

XG systems implement the abstract behaviors necessary to enable opportunistic use of spectrum, policy-defined operation, and traceability. As a guideline to identifying which abstract behaviors are necessary, we use the following rationale. XG systems must minimally agree on how they characterize spectrum conditions (spectral awareness) and on the opportunities available to use spectrum (opportunity instances), and must support behaviors that enable the dissemination of spectral awareness and opportunity information. In addition, for traceability, they must account for emissions they make in terms of the policies that authorize each emission, and the valid opportunities that were identified to enable the emission. Finally, they must support behaviors that allow systems to coordinate the use of opportunities either to form a data communications channel (at Layers 2 and above), or to avoid opportunities selected by other XG systems.

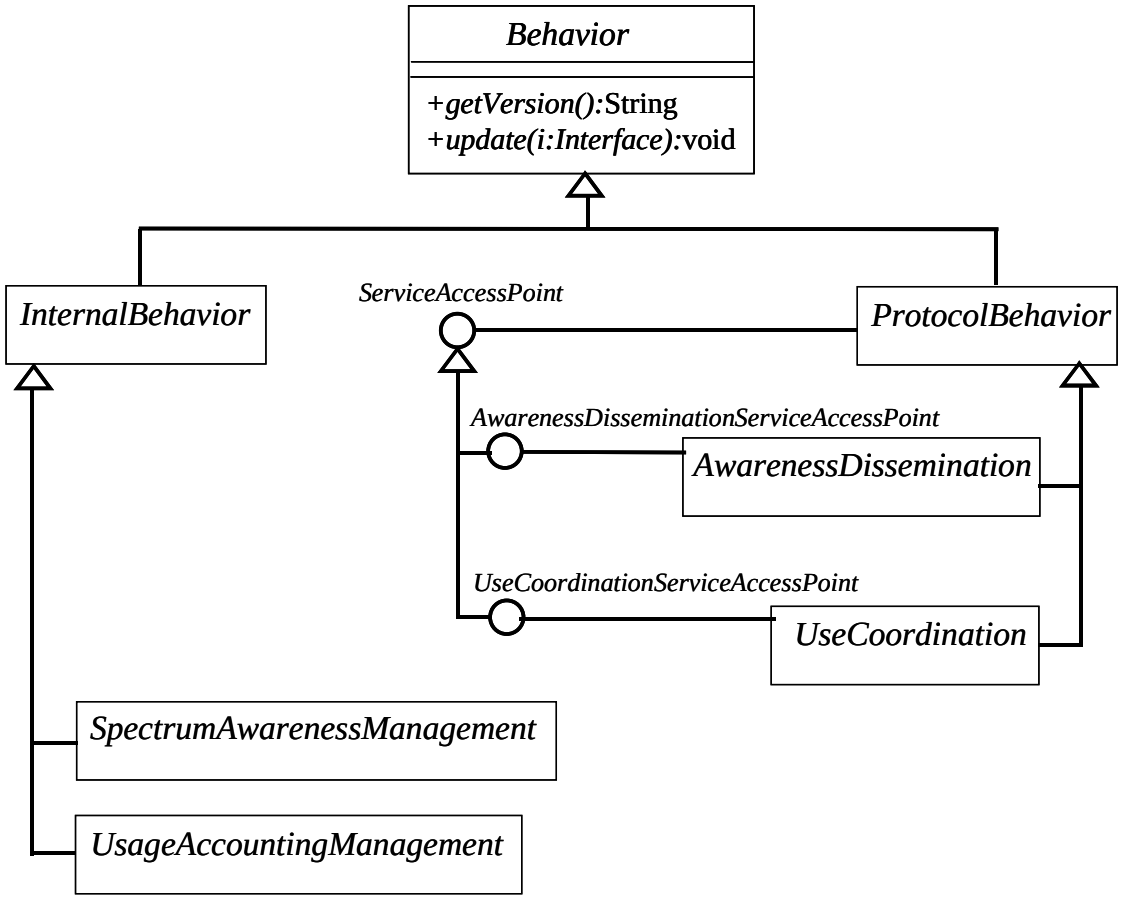


Figure 43: XG Internal and Protocol Behavior Classes

XG abstract behaviors extend the Behavior abstract class. As illustrated in Figure 43, the subclasses of Behavior include:

- InternalBehavior, the class of behaviors that are *internal* to the XG system
- ProtocolBehavior, the class of behaviors that involve communications with agents external to an XG system. The protocol behaviors use virtual coordination channels to exchange XG protocol information, and access to the protocol behaviors can be made through a ServiceAccessPoint interface.

Behaviors interact with other behaviors and the rest of the system through Interfaces. Behaviors and Interfaces implement the Observer-Subject design pattern. A behavior can register itself at an interface, and the behavior implements a generic *update* method that can be called by the interface to notify the behavior of any state changes.

We have identified the following four XG abstract behaviors:

1. **SpectrumAwarenessManagement:** an InternalBehavior, which describes how opportunity information is acquired, identified, represented and disseminated within and across XG systems. This behavior encompasses awareness information gained from sensing, configuration, and through AwarenessDissemination instances.
2. **AwarenessDissemination:** a ProtocolBehavior, which can be used by XG systems to share opportunity awareness information.
3. **UsageAccountingManagement:** an InternalBehavior, which enables emissions to be traced to a valid opportunity enabled by policy. This behavior is responsible for ensuring that every opportunity is validated for policy conformance prior to its use. This behavior is also responsible for ensuring that all parameters governing operation are correctly bound to the values specified within the validated opportunity.
4. **UseCoordination:** a ProtocolBehavior, which allows XG systems to coordinate the use of selected opportunities with other (XG and non-XG) systems.

A natural question to ask is why these four behaviors in particular are included in the accreditable kernel and why some others (notably, allocation) not included. Recall from Section 1 that abstract behaviors must satisfy at least one of the following to be included in the accreditable kernel:

- It is required to support opportunistic sharing of spectrum
- It is required to support policy-defined operation
- It provides traceability from policy through behavior to emissions

A radio without awareness of opportunities will cease to be an XG radio (i.e. it cannot perform opportunistic spectrum sharing); therefore, SpectrumAwarenessManagement is included within the accreditable kernel.

UsageAccountingManagement addresses the policy-defined operation and traceability requirements directly, and is included within the accreditable kernel.

In addition, traceability requires that trustable dissemination and coordination protocol behaviors (though not necessarily particular protocols) may be required by policy in some circumstances. An example of this requirement can be found in proposed approaches for the reuse of public safety bands through the use of beacon signals that authorize use. These two protocol behaviors (one for trusted dissemination of what's available to share and another for trusted signaling of opportunity acquisition and release) are therefore included in the accreditable kernel because they relate to policy conformance. Clearly, these behaviors are required only in those XG radios that make use of particular opportunities enabled by policy due to the implementation of these protocols.

Identification and allocation of opportunities need not be a trusted function. A tool for finding opportunities which devises wonderful spectrum use most of the time and occasionally suggests something infeasible/illegal is perfectly reasonable, provided that the infeasible opportunities are never instantiated. It is the barrier to instantiation that is the accreditable kernel's function – not the

exploration of opportunities. And leaving the exploration of opportunities outside the accreditable kernel allows for freer innovation.

We describe these behaviors in more detail in Section 0. The details of the architecture, design, and implementation of these behaviors and interfaces, however, are left open to innovation subject to policy, technology, and other business constraints.

2.4 XG Information Objects

We organize XG Information Objects into three abstract classes as illustrated in Figure 44: PolicyDefinedJoinPoint[11], Expression, and ProtocolDataUnit.

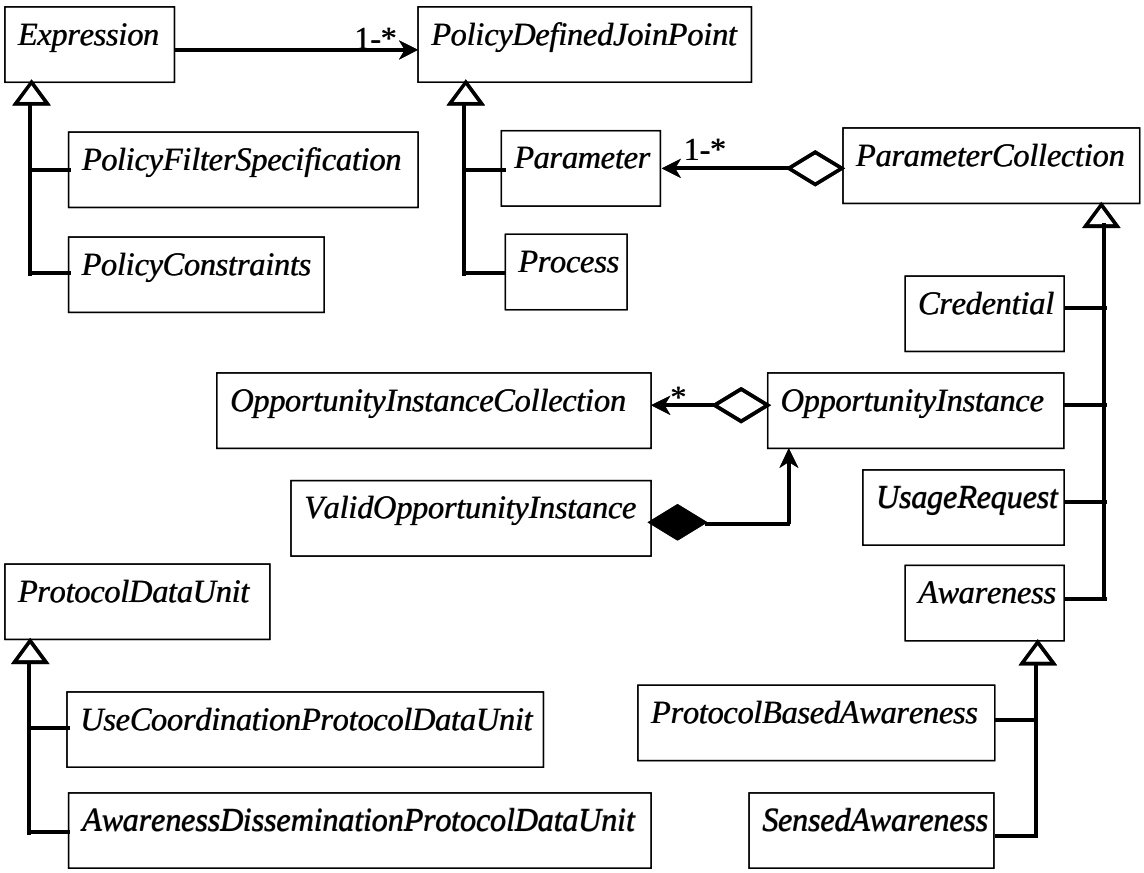


Figure 44: XG Information Objects and their associations

The PolicyDefinedJoinPoint, described further in Section 0 is a generalization of two abstract classes: Parameter and Process. Instances of the PolicyDefinedJoinPoint class are primitives that are described using the XG Policy Language Framework (see [XGPLF]), and implemented within the XG radio system. The AccreditableKernel through the XG System Capabilities Interface provides access to these primitives.

Instances of the Parameter class are parameters that govern the operation of the XG radio system, and instances of the Process class are methods, procedures, or relations that are implemented by the XG radio system.

The ParameterCollection is an abstract (and arbitrary) collection of Parameter instances. Subclasses of ParameterCollection include Credential, Awareness, UsageRequest, and OpportunityInstance.

[11] In computer science, a *join-point* is a point in the flow of a program. In Aspect Oriented Programming, a *pointcut* is a set of join-points. Whenever program execution reaches one of the join points defined in the *pointcut*, a piece of code (called advice) associated with the pointcut is executed. We envision a PolicyDefinedJoinPoint to be points within the XG radio where behavior governed by policy is accessible.

Instances of the Credential class encapsulate security and trust management information such as cryptographic keys and authentication information. Credential instances are exchanged across XG interfaces for trust management. The specification of the security and trust management architecture is not within the scope of this document.

The Awareness class encapsulates situational awareness information including spectral awareness, network awareness, and possibly temporal and geo-spatial information as well. Instances of the Awareness class are created and managed by the XG Spectrum Awareness Management behavior. Subclasses of the Awareness class include SensedAwareness that encapsulates awareness gained by the XG system through sensing, and ProtocolBasedAwareness, which encapsulates awareness acquired by the XG system through the use of awareness dissemination protocols.

The UsageRequest class and OpportunityInstance class respectively encapsulate information in the request for, and in the characterization of, spectrum use. OpportunityInstanceCollection is an abstract collection of instances of the OpportunityInstance class. Instances of UsageRequest can be sent (to the SystemStrategyReasoner, for example) over the AllocationInterface, and an OpportunityInstanceCollection that includes a list of opportunities may be returned. The radio system can choose opportunities from this collection, and then pass the selected instance of OpportunityInstance over the PolicyInterface to have it validated (i.e. checked whether it is authorized by applicable policy). A ValidOpportunityInstance object encapsulates validation status and associated credential information of an OpportunityInstance object that it contains. The XG Usage Accounting Management behavior ensures that operation (e.g. emission) conforms to the parameter values set within the ValidOpportunityInstance with the appropriate status and credentials returned by the PolicyInterface.

The Expression abstract class encapsulates predicate expressions that involve PolicyDefinedJoinPoint instances. The expression language can be specific to the particular XG implementation. Subclasses of the Expression class include the PolicyFilterSpecification class, which encapsulates expressions that specify a filter on policy instances based on arbitrary criteria of interest, and the PolicyConstraints class, which encapsulates expressions that specify constraints that apply to the XG radio system. These expressions can be, for example, used by the SystemStrategyReasoner to interact with the PolicyConformanceReasoner.

The ProtocolDataUnit is an abstract class that encapsulates information exchanged across ServiceAccessPoint interfaces (and any attributes of that information, required to effect an exchange). In particular, two subclasses of ProtocolDataUnit, namely, the AwarenessDisseminationProtocolDataUnit and UseCoordinationProtocolDataUnit are exchanged over the VirtualCoordinationChannelInterface by instances of the XG protocol behaviors (AwarenessDissemination and UseCoordination). The extension and implementation of these abstract classes are specific to the protocols used by the particular XG radio system.

2.5 XG System Interactions

There are three key system interactions that relate to opportunity awareness, allocation, and use.

The SpectrumAwarenessManagement behavior creates and manages Awareness objects, objects that provide information about the environment around the XG radio. To get the information necessary to create, update, modify, or delete Awareness objects, the behavior acquires SensedAwareness objects via the SensorInterface and ProtocolBasedAwareness objects from the AwarenessDisseminationServiceAccessPoint interface. The latter interface is implemented by the AwarenessDissemination behavior, which exchanges spectrum information (abstractly modeled as AwarenessDisseminationProtocolDataUnit instances) with other systems via communications channels modeled by the VirtualCoordinationChannelInterface. These interactions are illustrated in Figure 45.

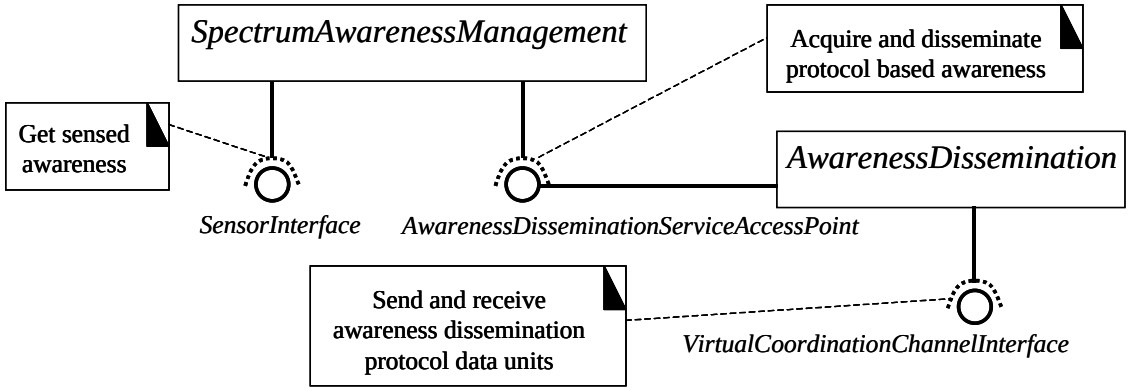


Figure 45: Spectrum awareness is gained through sensing and protocol behaviors

Once awareness of opportunities is acquired, the next step is to identify and allocate opportunities for use by the radio system under policy constraints. As illustrated in Figure 46, the SystemStrategyReasoner performs this function. The SystemStrategyReasoner can access policy information through the PolicyInterface, and it can access the capabilities, configuration, state, and primitives within the radio system through the SystemCapabilitiesInterface. In particular, this includes access to the XG SpectrumAwarenessManagement behavior.

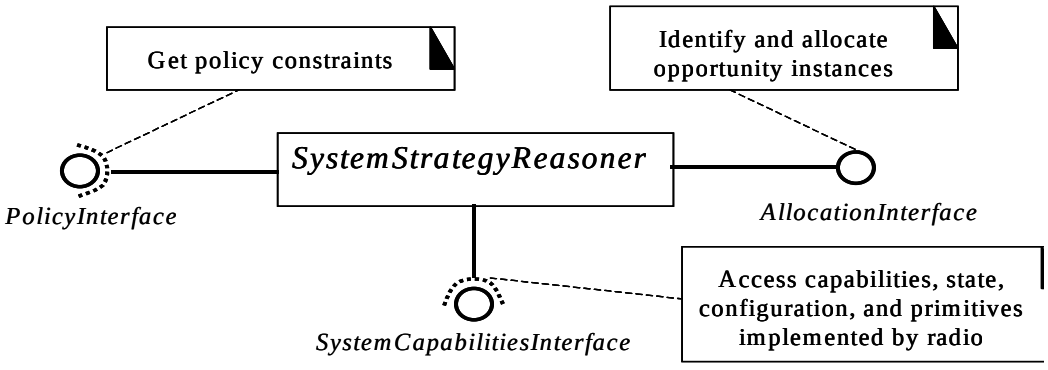


Figure 46: The System Strategy Reasoner identifies and allocates Opportunity Instances for the Radio; it may additionally access radio primitives to modify system behavior

Once opportunities have been identified and allocated based on awareness of opportunities, one or more can be selected and used in a manner that is authorized by policy. The UsageAccountingManagement behavior performs the function of ensuring that OpportunityInstance objects are presented to the PolicyInterface prior to use, and that only ValidOpportunityInstance objects returned by the PolicyInterface with proper validation status and credentials are actually used.

The UsageAccountingManagement behavior through the SystemCapabilitiesInterface asserts that subsystems actually use the parameter values contained within the ValidOpportunityInstance, before providing it to the TransceiverInterface. Furthermore, authorized use of spectrum may entail the employment of a protocol to coordinate with other nodes. For this purpose, the UsageAccountingManagement behavior interacts with the UseCoordination protocol behavior through the UseCoordinationServiceAccessPoint.

The interactions related to the use of opportunities are illustrated in Figure 47.

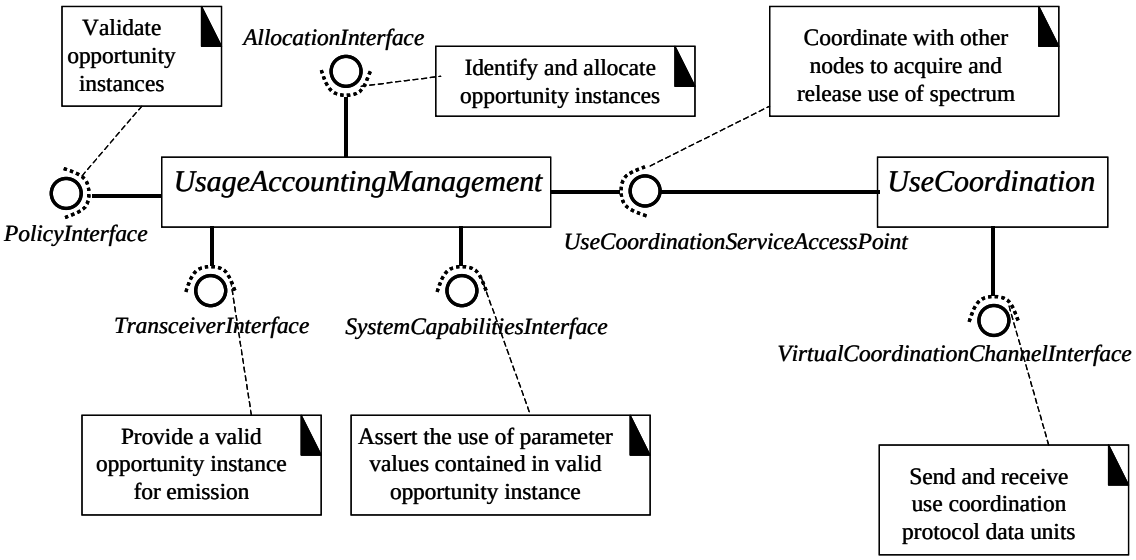


Figure 47: Opportunity instances are allocated, validated against policy, then used after asserting that all parameter values are set appropriately and after coordination with peers

3 XG Interfaces

In this section, we describe the following XG abstract interfaces:

- **SensorInterface**
- **TransceiverInterface**
- **SystemCapabilitiesInterface**
- **PolicyInterface**
- **AllocationInterface**
- **VirtualCoordinationChannelInterface**
- **XgToMediumAccessControlInterface**
- **AwarenessDisseminationServiceAccessPoint**
- **UseCoordinationServiceAccessPoint**

For each interface, we discuss its methods and the semantics of those methods (what the interface seeks to do).

We also discuss common error situations for each interface. Because this document does not specify how errors are handled (whether by exceptions or return values or some other means) the error discussion is intended simply to highlight errors that must be addressed.

3.1 SensorInterface

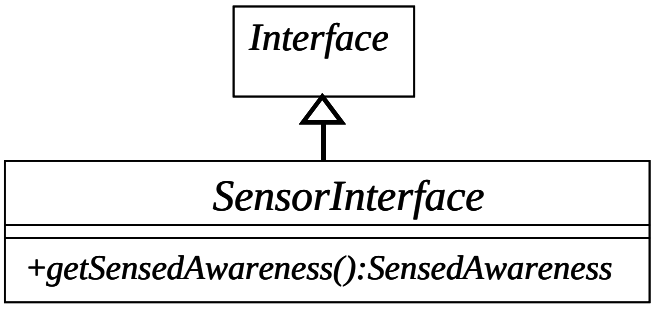


Figure 48: SensorInterface methods

The SensorInterface provides methods to enable sensed awareness to be requested and received from a sensor. The information that has been sensed is accessed through the returned SensedAwareness object(s). The SensedAwareness class is described later in Section 0.

The SensorInterface extends Interface; therefore, it includes the *getVersion*, *getCredential*, *addObserver*, *removeObserver*, and *notifyObservers* methods. A behavior, such as an instance of SpectrumAwarenessManagement, can register itself at a SensorInterface, be notified of changes to sensed information, and obtain *SensedAwareness* objects through the *getSensedAwareness* method.

We note here that sensor primitives (e.g. parameters such as look-through interval, integration time and frequency resolution) can be accessed through the SystemCapabilitiesInterface. Particular implementations may choose to provide access to these parameters by extending the SensorInterface and implementing additional methods.

Implementations can extend this interface further, for example, to provide more advanced control over the sensor tasking. For example, extensions can include methods to tailor sensing requests by providing a sensor mask that specifies the information to include in the response, and a notification condition expression that determines when (and how often) to provide notification. Whenever the notification condition is true (having fulfilled constraints and having applied the sensor mask) this method will provide sensed awareness by calling the invoker's *notifyObservers* method.

Error Conditions: There are no important error conditions to worry about in this interface. The only challenging case is how to handle cases where no sensed data can be made available, due to configuration errors, sensor failure, and the like. The expectation, in this interface, is that the result is a particular *SensedAwareness* object that indicates a lack of data or inability to get it.

3.2 TransceiverInterface

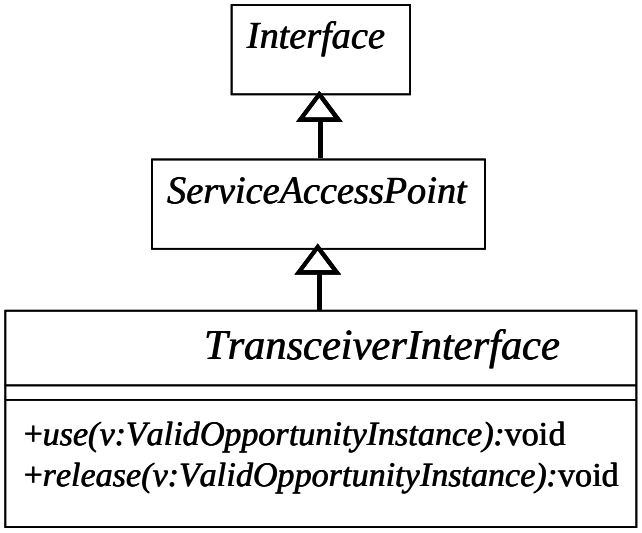


Figure 49: Transceiver Interface methods

Recall that the ServiceAccessPoint interface extends the basic Interface to support *send* and *receive* methods. The TransceiverInterface in turn extends ServiceAccessPoint to specify the opportunity (the ValidOpportunityInstance object) to use when sending and receiving instances of the ProtocolDataUnit object. (Recall that Interface, in this context, is not a network interface but a class definition).

The interface also provides a method to stop using an opportunity, through the *release* method.

We note here that transceiver primitives can be accessed uniformly through the SystemCapabilitiesInterface. Particular implementations can, however, extend the TransceiverInterface to provide more advanced control over transceiver operation or to provide access to its parameters.

Error Conditions: The important error conditions in this interface involve situations where the *ValidOpportunityInstance* is either, not usable from the start, or becomes unusable.

In general, because *ValidOpportunityInstance* objects are generated and/or verified by policy software, the expectation is that most errors will take place at the time of sending, when *send* is invoked. So a perfectly reasonable error handling approach is to handle all errors related to *ValidOpportunityInstance* at the time of sending. Such error handling would include checking that the opportunity is indeed authorized, and that the conditions under which the opportunity is valid still obtained (or, by changing radio configuration, can be restored).

3.3 SystemCapabilitiesInterface

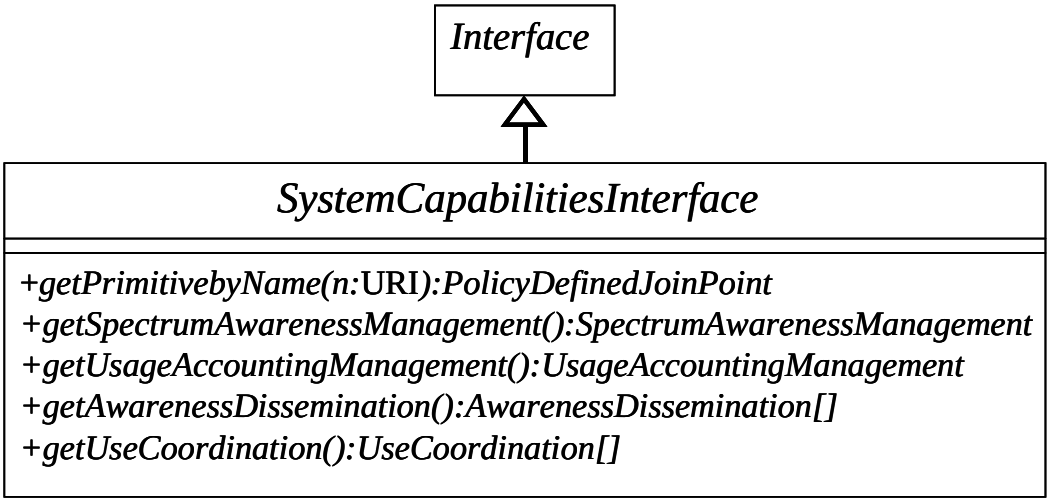


Figure 49: SystemCapabilitiesInterface methods

The SystemCapabilitiesInterface provides access to the capabilities, the configuration, and the state of the XG radio system. XG system capabilities include sensing capabilities (e.g., spectral power characterization, specific signal identification, and location sensing), transceiver capabilities (e.g., frequency agility, power control, waveform agility, and beam-forming), and other system capabilities (such as encryption support, and emulation of legacy PHY and MAC layers).

As envisioned in the XG Policy Language Framework RFC, access to XG system capabilities, configuration, and state are provided through primitives (*Parameter* and *Process*) described in a common Web ontology, and therefore, have an associated Web Universal Resource Identifier (URI). Policies are expressed in terms of these primitives that are implemented within various subsystems of the XG radio systems. As XG technology evolves, new systems with increasingly sophisticated capabilities will be developed, and therefore the SystemCapabilitiesInterface and its associated classes must be extensible.

Central to the SystemCapabilitiesInterface is the *getPrimitivebyName* method, which provides access to the primitives implemented in the radio in terms of their URI by returning a PolicyDefinedJoinPoint instance. The PolicyDefinedJoinPoint class, which we describe in Section 25.1, generalizes the Parameter and Process classes, and enables the SystemCapabilitiesInterface remain extensible to future developments.

In addition, four utility methods to access the implemented behaviors – g*etSpectrumAwarenessManagement, getAwarenessDissemination, getUseCoordination, and getUsageAccountingManagement* – are provided by SystemCapabilitiesInterface.

The SystemCapabilitiesInterface follows the Singleton and Facade design patterns [DP]: it serves as a one-stop shop for a SystemStrategyReasoner instance to register itself (through the Observer pattern methods inherited from Interface), and access primitives and key behaviors implemented within various subsystems of the XG radio system.

Error Conditions: Obviously, there is a range of possible errors involving URIs. For instance, the specified capability may not be present or the URI may be malformed. Errors can also result if the implementation of the accessed primitive has an internal error condition.

3.4 PolicyInterface

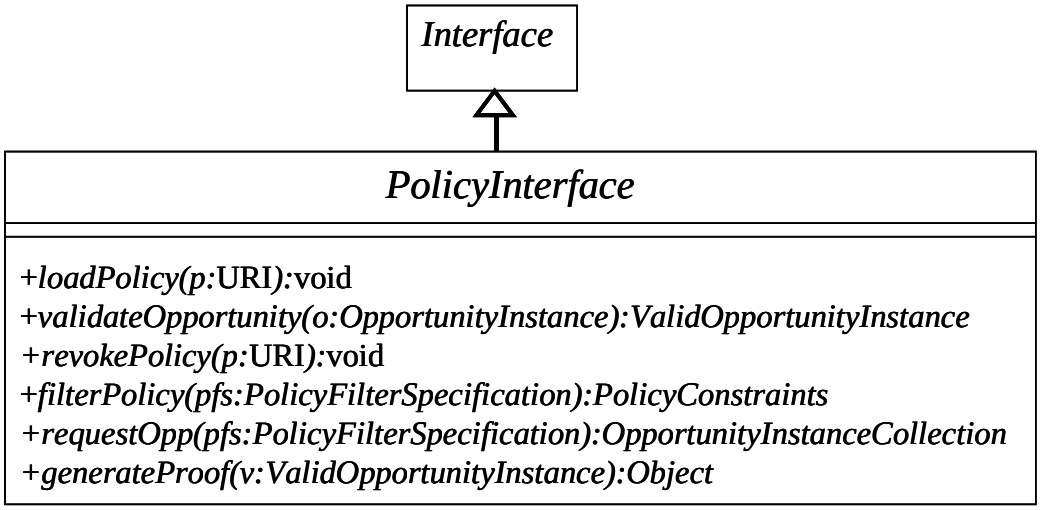


Figure 50: PolicyInterface methods

The PolicyInterface provides a *validateOpportunity* method to validate the conformance of proposed spectrum usage to applicable policies. It accepts an OpportunityInstance object that contains attribute-value pairs that fully characterize the proposed spectrum use, and returns a ValidOpportunityInstance object that indicates whether the proposed spectrum use conforms to policy and includes credentials from the PolicyInterface. The UsageAccountingManagement behavior can propose an OpportunityInstance object through this method, and then assert various system parameters based on the returned ValidOpportunityInstance object. These information objects are described further in Section 25.3.

The PolicyInterface also provides five additional methods:

1. *loadPolicy* to retrieve and load machine-understandable policy sets from a repository
2. *revokePolicy* to convey the revocation of loaded policy sets
3. *filterPolicy* to extract policies that apply to a particular situation
4. *requestOpp* that searches for opportunities that are authorized by policy
5. *generateProof* that provides a machine proof that a given ValidOpportunityInstance conforms to applicable policy

The *filterPolicy* method accepts a *PolicyFilterSpecification* object and returns a *PolicyConstraints* object. The *PolicyFilterSpecification* object can be based upon a particular situation (e.g. location, frequency bands, time) or a particular device description (e.g. transceiver tuning range, capable of beamforming, supports morphed waveforms, particular level of certification). The returned *PolicyConstraints* object can be used, for example, by a SystemStrategyReasoner instance to reason about policy and identify opportunities in the context of the particular system. The design of the *PolicyFilterSpecification* and *PolicyConstraints* objects is specific to particular implementations.

The *requestOpp* method is used to search for opportunities that are allowed by policy without considering particular requirements of the radio. The returned list of opportunities may be further constrained or a subset selected from the list by a system-specific allocation procedure.

The *generateProof* method outputs a machine proof – a sequence of assertions deduced using policy-processing rules and based on awareness and policy axioms – that a given opportunity conforms to or violates applicable policy. Such a proof can be used for diagnostic purposes and the format of the machine proof is implementation specific.

Error Conditions: Error conditions can result during loading of policy if the encoded policy is either unavailable or inaccessible. Furthermore syntactic or logical inconsistencies resulting from loading or revocation of policies can also result in error conditions. Malformed policy filter specifications can also result in errors.

3.5 AllocationInterface

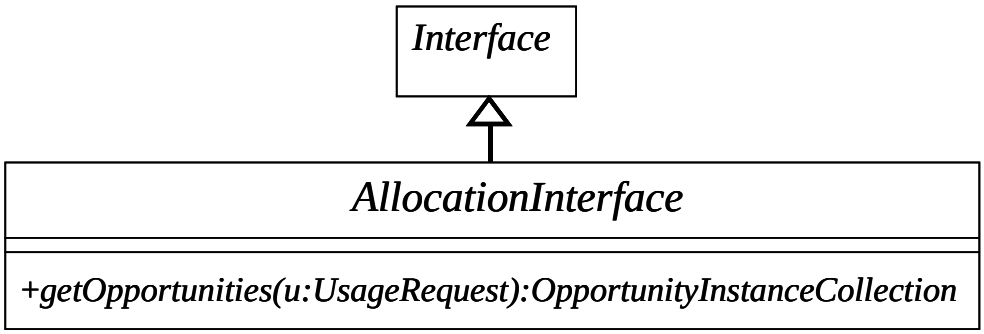


Figure 51: AllocationInterface methods

The Allocation Interface provides a method called *getOpportunities* that takes as input a UsageRequest object and returns an OpportunityInstanceCollection object. The class that implements this interface must therefore perform the function of identifying and allocating opportunities that are suitable for the XG radio system in a particular situation. In contrast to the PolicyInterface, which only validates whether an opportunity conforms to policy, the AllocationInterface provides a list of suitable opportunities for the radio system.

For example, in our notional system model, the SystemStrategyReasoner class performs the allocation procedure and implements the AllocationInterface. In some XG systems, the MediumAccessControl may perform the allocation procedure and this subsystem may implement the AllocationInterface in addition to the XgToMediumAccessControlInterface. The UsageAccountingManagement behavior can access this interface to obtain opportunities, chooses one or more opportunities, and then presents them to the PolicyInterface for validation prior to using those opportunities.

The problem of identifying and allocating opportunities among nodes is closely related to the coordination among various nodes that must share spectrum. The class that implements the allocation procedure (and the AllocationInterface) must therefore ensure that: (i) unrelated transmissions do not interfere with each other beyond what is allowed by policy, and (ii) nodes that wish to form a link are able to do so, for example, the transmitter and the receivers that are involved in a communication are tuned to the same opportunity. The UseCoordination behavior described later in Section 0 provides a mechanism for XG systems to share this allocation.

Error Conditions: Malformed usage requests can result in errors.

3.6 VirtualCoordinationChannelInterface

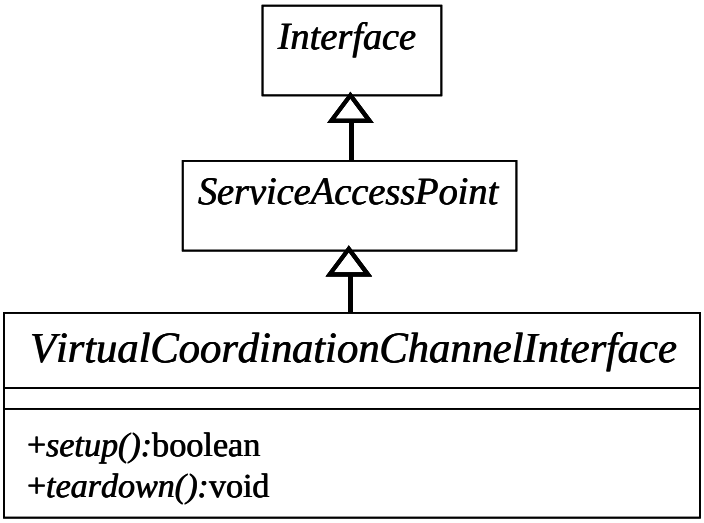


Figure 52: VirtualCoordinationChannelInterface methods

The VirtualCoordinationChannelInterface provides methods to configure and access logical control channels for use by the XG protocol behaviors. At a minimum, logical coordination channels can be created and destroyed using the *setup* and *teardown* methods. (The assumption is that the radio knows how the control channels are to be created/destroyed and this interface simply says "do it"). This interface inherits from the ServiceAccessPoint interface two methods – *send* and *receive* – that can be used by the protocol behaviors to exchange ProtocolDataUnit objects.

Virtual coordination channel primitives can be accessed uniformly through the SystemCapabilitiesInterface. Particular implementations can, however, extend the VirtualCoordinationChannelInterface to provide more advanced control over the operation of a virtual coordination channel or to provide access to its parameters.

Error Conditions: There's one clear error case: that the radio system is not configured to have a coordination channel in its current operating mode. This error can either be detected and signaled as a result of invoking *setup*, or when the channel is first used by *send* or *receive*.

3.7 XgToMediumAccessControlInterface

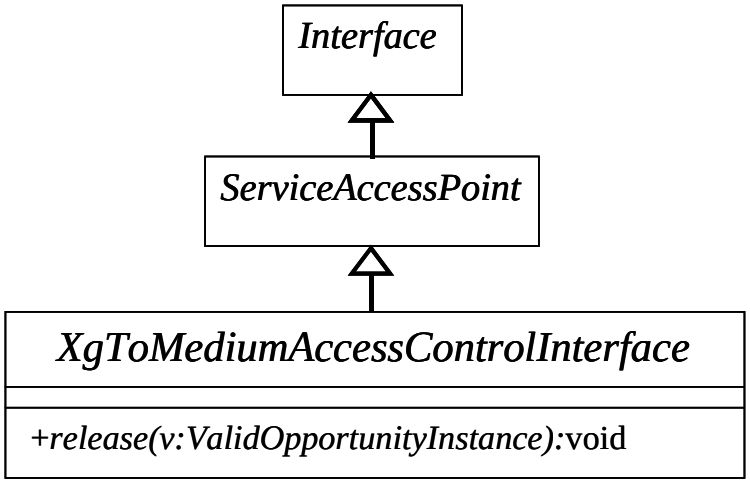


Figure 53: XgToMediumAccessControlInterface methods

The XgToMediumAccessControlInterface provides access methods to coordinate XG operation with the medium access control layer:

1. Exchange ProtocolDataUnit objects (to be passed to the transceiver interface) through the *send* and *receive* methods inherited from ServiceAccessPoint
2. Notify the medium access control layer when the transceiver is ready to send or receive frames (e.g. after any transceiver reconfiguration that was requested is completed) through the *notifyObservers* method inherited from Interface

One additional method *release* is defined for the XgToMediumAccessControlInterface. This method may be called by UsageAccountingManagement to notify an XG-aware MediumAccessControl instance of the expiration or invalidation of a previously requested opportunity.

Medium access control primitives (parameters) can be accessed uniformly through the SystemCapabilitiesInterface. Particular implementations can, however, extend the XgToMediumAccessControlInterface to provide more advanced control over medium access control operation or to provide access to its parameters.

Error Conditions: Calling the release method on a legacy MediumAccessControl instance (that is not XG-aware) is an error. Calling the release method for a ValidOpportunityInstance not already known to the MediumAccessControl is another potential error condition. Implementations must include suitable mechanisms to handle these error conditions.

3.8 AwarenessDisseminationServiceAccessPoint

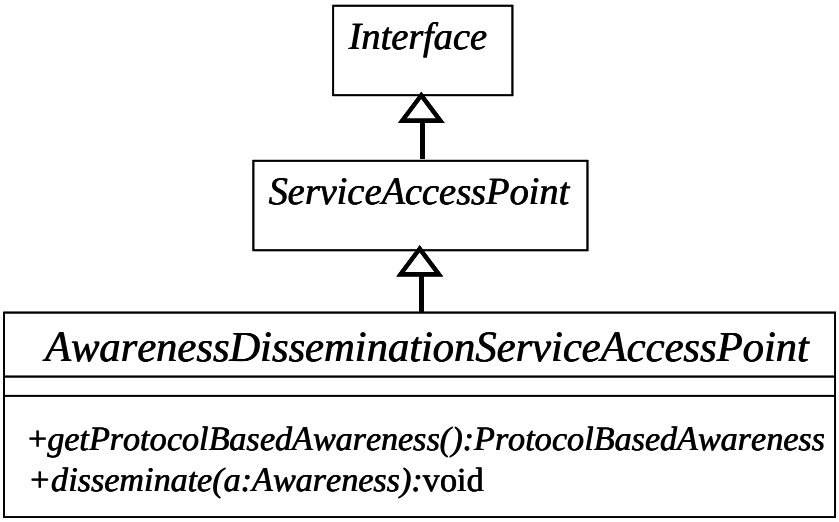


Figure 54: AwarenessDisseminationServiceAccessPoint methods

The AwarenessDisseminationServiceAccessPoint provides methods to enable disseminated awareness to be requested and received from other XG nodes. An instance (e.g., of *SpectrumAwarenessManagement*) can register itself at an AwarenessDisseminationServiceAccessPoint, be notified of changes to disseminated awareness information, and can obtain *ProtocolBasedAwareness* objects through the *getProtocolBasedAwareness* method. The ProtocolBasedAwareness class is described later in Section 25.2.

The AwarenessDisseminationServiceAccessPoint also implements a *disseminate* method to enable requests to distribute spectral awareness information.

Obvious possible extensions to this interface include facilities to filter the information returned in *ProtocolBasedAwareness* objects, or to limit when *notifyObserver* is invoked. Such extensions could be used to limit the nodes with which awareness is shared (or received), or to limit the frequency bands about which an instance wishes to be kept aware, or the frequency of information updates.

Error Conditions: The absence of awareness is not an error. Communications outages and the like can cause this condition and systems must deal with it gracefully. The inability to disseminate information is also not an error (for similar reasons), however, there must be some way for the system to discover its inability to disseminate.

3.9 UseCoordinationServiceAccessPoint

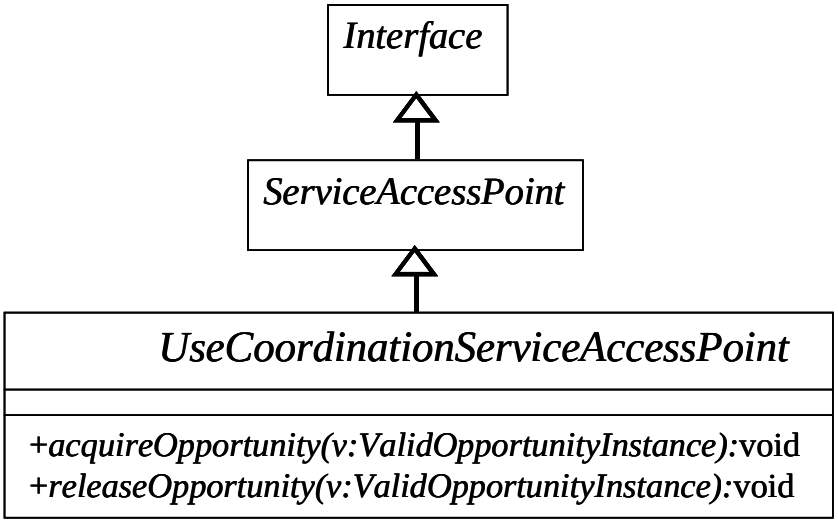


Figure 55: UseCoordinationServiceAccessPoint methods

The UseCoordinationServiceAccessPoint provides the *acquireOpportunity* and the *releaseOpportunity* methods to signal the acquisition and release of opportunities to external devices (e.g. other XG radios) in a manner that conforms to policy. The UsageAccountingManagement method can pass relevant parameters to this interface through a ValidOpportunityInstance object.

The difference between this interface and TransceiverInterface is that TransceiverInterface purely affects internal workings of the radio, while UseCoordinationServiceAccessPoint involves coordinating use with external radios. To implement an opportunity, in many cases, will require using both interfaces (to configure the radio and to coordinate that configuration with other radios).

Error Conditions: All the error conditions that TransceiverInterface must worry about are relevant here as well. In addition, attempts to coordinate with other radios may fail or still be underway at a time when an attempt is made to utilize the opportunity. Care in designing the error interface is called for, as errors may be transient (e.g., negotiations in progress) rather than permanent.

4 XG Behaviors

In this section, we will describe the following four XG abstract behaviors:

- **SpectrumAwarenessManagement**
- **Awareness Dissemination**
- **UsageAccountingManagement**
- **UseCoordination**

In the following subsections, we describe these behaviors, which are essential in order to ensure that (i) spectral awareness information is acquired and disseminated in order to make adaptive use of available spectrum, and (ii) such use is made in a manner that conforms to policy and is coordinated with other nodes as required.

4.1 SpectrumAwarenessManagement

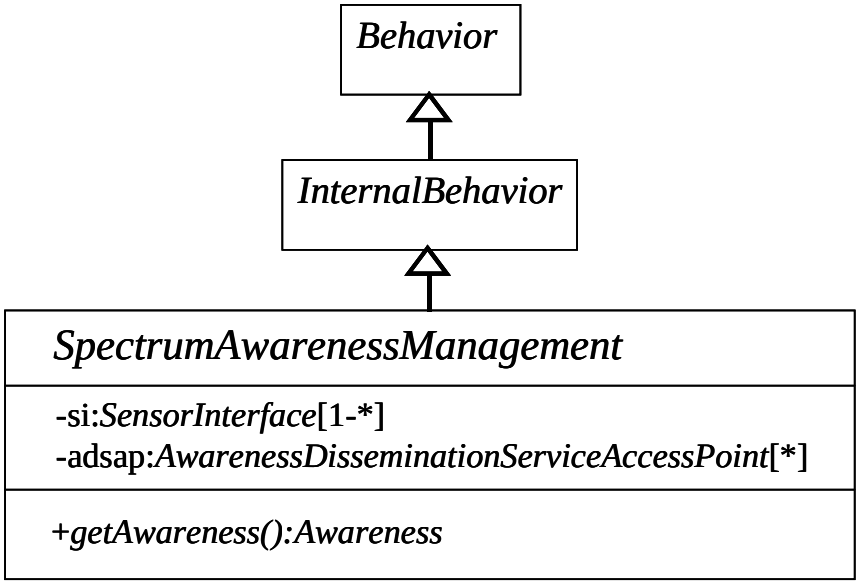


Figure 56: SpectrumAwarenessManagement class

Central to the idea of a XG radio is spectrum awareness. This awareness can come from a number of sources. The purpose of the *SpectrumAwarenessManagemen*t behavior is to provide a single trusted entity that distributes, manages, consolidates and disseminates spectrum information.

SpectrumAwarenessManagement manages spectrum awareness from SensedAwareness acquired through the SensorInterface and from ProtocolBasedAwareness acquired through the AwarenessDisseminationServiceAccessPoint. In turn, SpectrumAwarenessManagement provides access to its acquired awareness through the *getAwareness* method, which returns an instance of the Awareness class, described further in Section 0. The information provided by this behavior can be used by other behaviors in order to make decisions regarding opportunity identification, allocation, and dissemination.

Note that as part of its role in managing awareness, SpectrumAwarenessManagement may edit, filter or combine awareness objects. (For instance, if the radio is only interested in information about television frequency bands, it may edit all awareness information to only contain information about these bands). It is the ability to edit and alter which makes it essential that SpectrumAwarenessManagement be a trusted piece of the accreditable kernel. (Note that, even though trusted, the behavior is expected to track where information is learned, for purposes of audit and verification).

The SpectrumAwarenessManagement behavior registers itself to receive notifications from the SensorInterface and AwarenessDisseminationServiceAccessPoint through the respective *addObserver* methods. As a result, the *update* method will be invoked whenever new sensed or protocol-based awareness information is received. The SpectrumAwarenessManagement behavior must then call the *getSensedAwareness* or the *getProtocolBasedAwareness* method on the respective interface and process the awareness information returned to form a common picture. The behavior may also call *disseminate* to distribute its spectrum awareness to others.

A notional sequence of operations for the SpectrumAwarenessManagement behavior is illustrated in Figure 57.

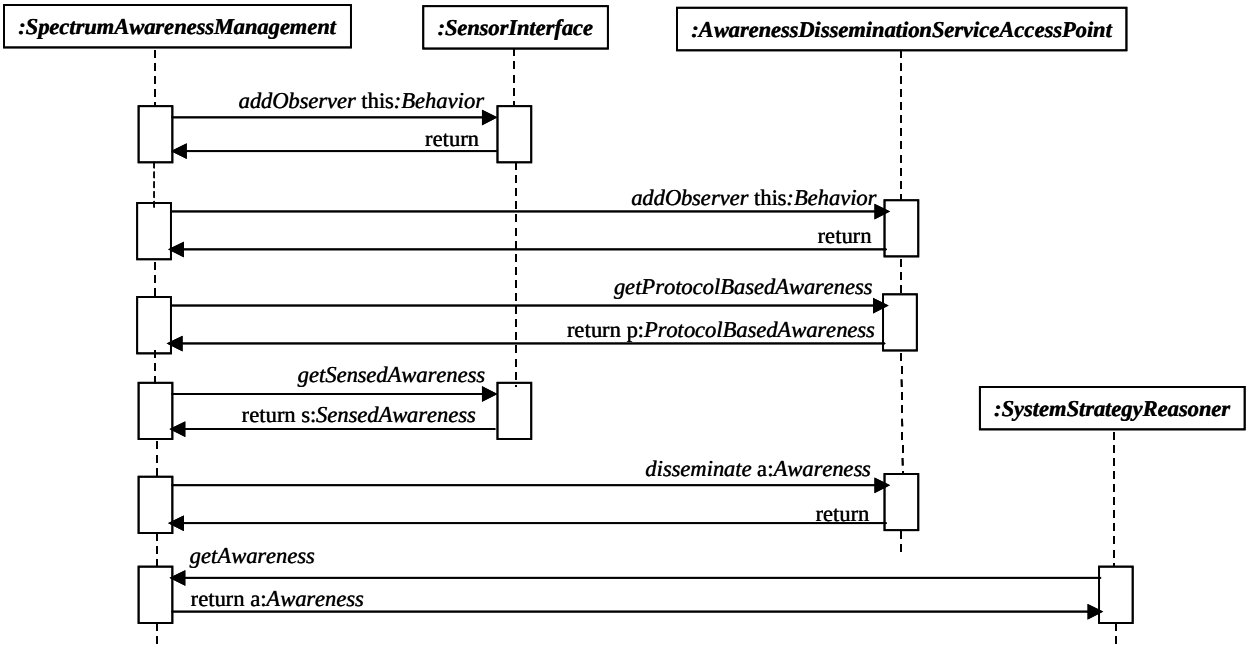


Figure 57: A notional sequence diagram for the SpectrumAwarenessManagement behavior

4.2 AwarenessDissemination

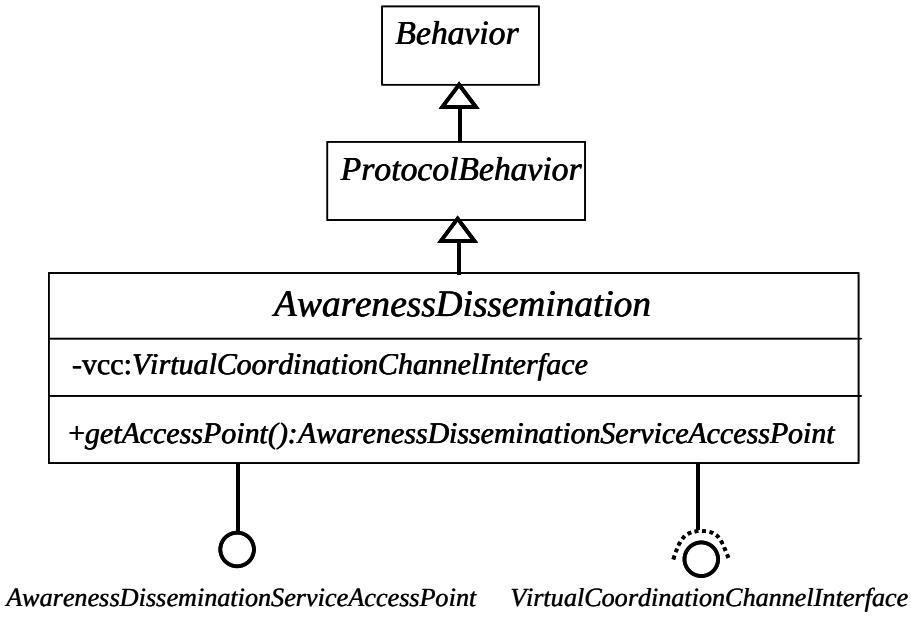


Figure 58: AwarenessDissemination class

The AwarenessDissemination protocol behavior allows XG systems to acquire protocol-based awareness. As part of this awareness, this behavior must discover neighboring systems that can be reached through virtual coordination channels. AwarenessDissemination implements the AwarenessDisseminationServiceAccessPoint interface, and is associated with an instance of the VirtualCoordinationChannelInterface. AwarenessDissemination implements the *getAccessPoint* method, which returns the AwarenessDisseminationServiceAccessPoint

AwarenessDissemination protocols can have diverse implementations. The protocols may be centralized, decentralized, or distributed. Furthermore, they can be implemented at different networking layers depending on how the virtual coordination control channel is implemented. For example, an interruptible spectrum model that permits reuse of public-safety channels through a use-permitted beacon signal is an example of a centralized, 1-hop, physical-layer AwarenessDissemination subclass. A taxonomy of AwarenessDissemination protocols is shown in Figure 59.

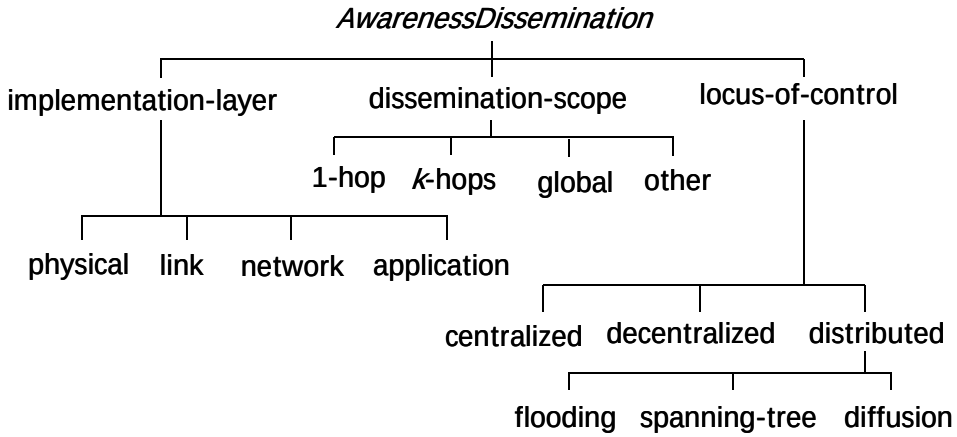


Figure 59: Awareness Dissemination Taxonomy

4.3 UsageAccountingManagement

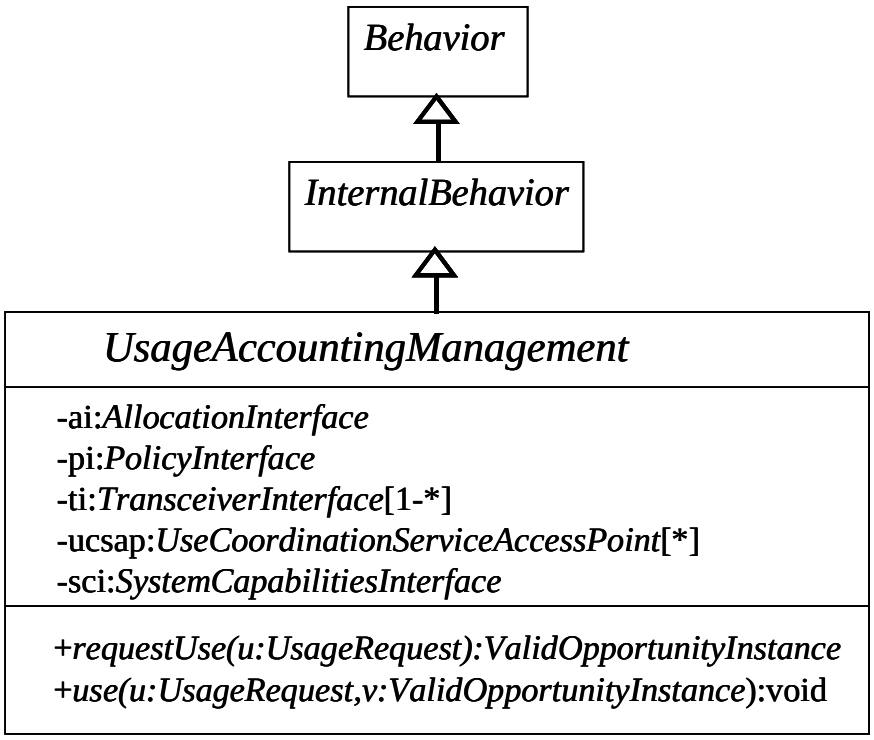


Figure 60: UsageAccountingManagement class

The UsageAccountingManagement behavior is responsible for ensuring that opportunities are validated though the PolicyInterface prior to their use. This behavior is also responsible for ensuring that parameters governing operation are bound to the values specified within the ValidOpportunityInstance object to be used.

Based on either a potential or an actual requirement for spectrum use within the radio system, the UsageAccountingManagement behavior requests opportunity allocations by providing a UsageRequest object to the AllocationInterface. In response, the AllocationInterface returns an OpportunityInstanceCollection object that contains opportunity allocations. The UsageAccountingManagement behavior presents one or more of the allocated opportunities to the PolicyInterface for validation in order to obtain a ValidOpportunityInstance object.

Upon receiving a request for spectrum use, the UsageAccountingManagement behavior uses the ValidOpportunityInstance object resulting from validation to assert parameters governing spectrum use through the SystemCapabilitiesInterface. This behavior also invokes the *acquireOpportunity* and *releaseOpportunity* methods on the UseCoordinationServiceAccessPoint if policy requires coordination of use through protocols. UsageAccountingManagement therefore maintains needed associations to the five interfaces mentioned above.

The UsageAccountingManagement behavior is key to realizing the traceability goals of the XG program. XG systems must account for emissions they make, which means that they should characterize the emission, and also include information about the specific opportunity identified for this use including the source of this opportunity (sensing, policy, or learned from an external source though a protocol), as well as information about policy that permits such usage. Such information is contained within the ValidOpportunityInstance object described later in Section 25.3.

The UsageAccountingManagement behavior provides a *use* method that:

- Associates a UsageRequest with the corresponding ValidOpportunityInstance that was used
- Asserts system parameters are set to the values specified in the ValidOpportunityInstance
- Calls the *use* method of the TransceiverInterface, and if needed, the *acquireOpportunity* and *releaseOpportunity* methods of the UseCoordinationServiceAccessPoint.

The decoupling of the requestUse and use methods is worth noting – the former ensures that an opportunity is validated by the policy conformance reasoner prior to use, and the latter method ensures that all parameters are actually bound to the values contained in the validated opportunity. This enables the XG radio to use a validated opportunity just once or multiple times as long as all parameters can be asserted to have the appropriate values. Opportunity expiration can be controlled by any number of parameters, including possibly, explicit parameters for the maximum reuse count before validation or an opportunity expiration time.

A notional sequence of operations for the UsageAccountingManagement behavior is illustrated in Figure 60. In the case of XG-aware MediumAccessControl instances, the release method of the XgToMediumAccessControlInterface may also be invoked (not shown in Figure 61).

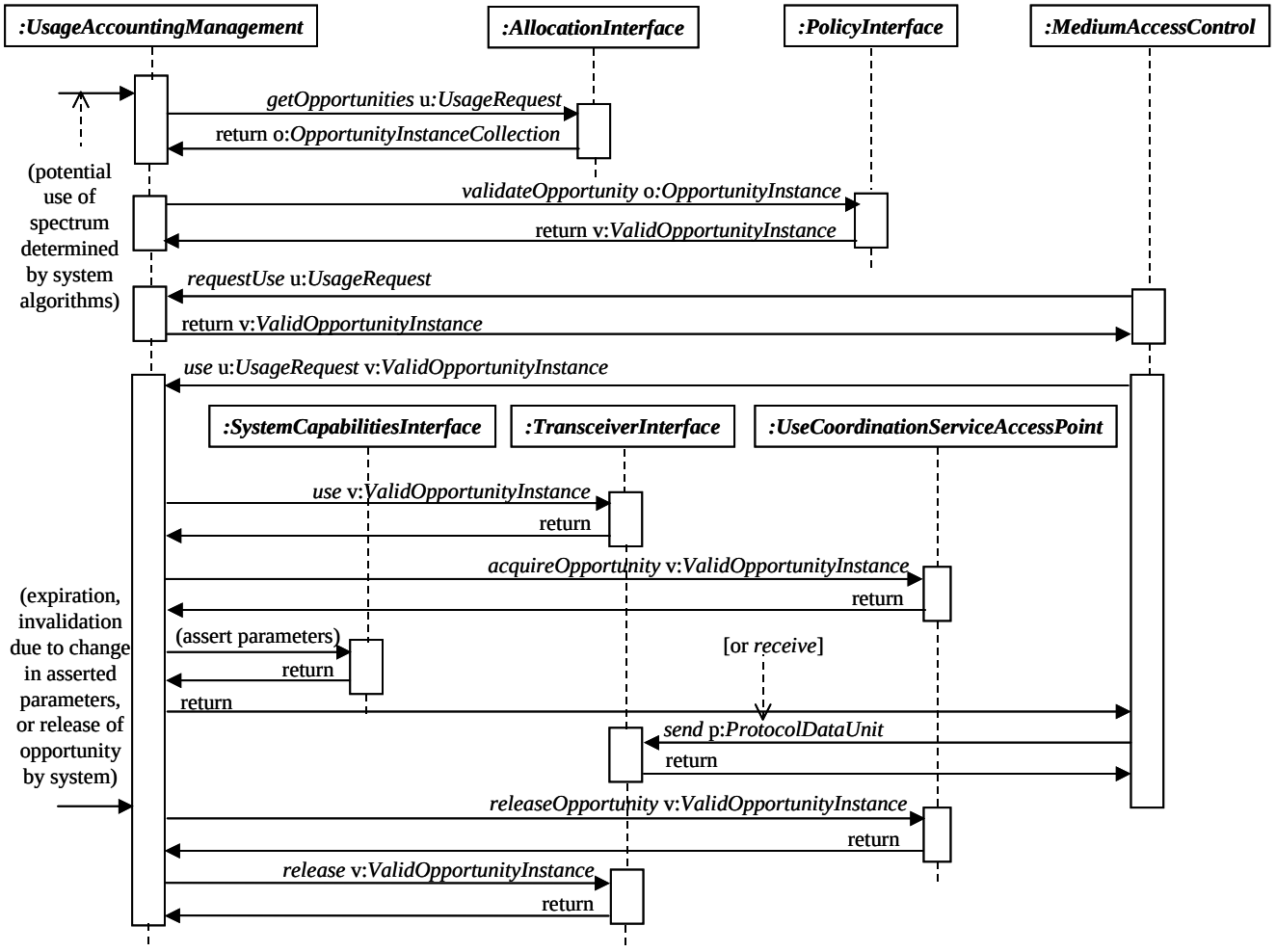


Figure 61: A notional sequence diagram for the UsageAccountingManagement behavior

4.4 UseCoordination

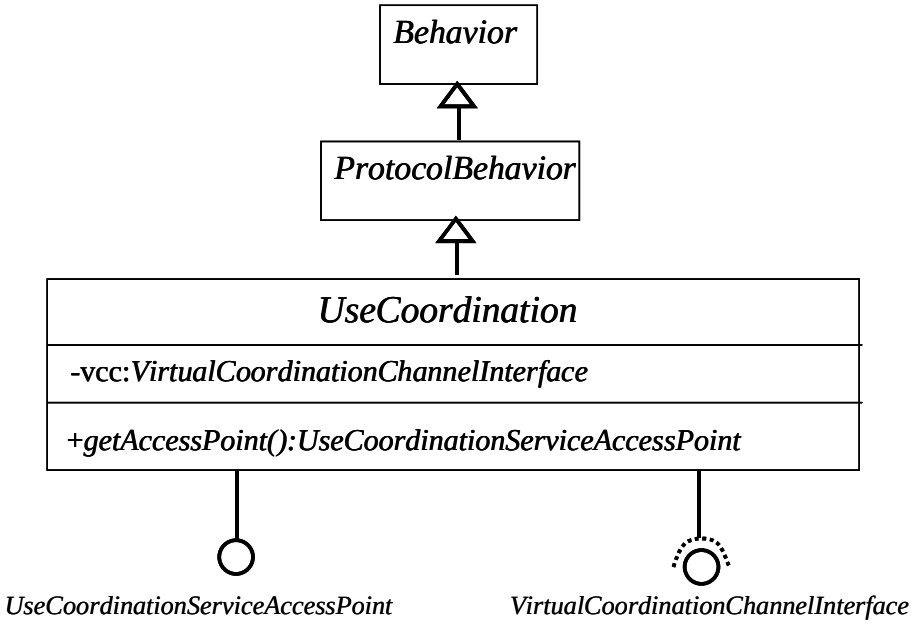


Figure 62: UseCoordination class

The UseCoordination protocol behavior communicates the specific opportunity selected for use by an XG system. This behavior enables XG systems to coordinate the use of selected opportunities with other systems (both XG and non-XG), including *acquisition* and *release* of the opportunity. UseCoordination implements the UseCoordinationServiceAccessPoint interface, and is associated with an instance of the VirtualCoordinationChannelInterface. UseCoordination implements the *getAccessPoint* method, which returns the UseCoordinationServiceAccessPoint

The XG-UCP can be used under two distinct settings:

- XG systems that wish to communicate with each other (i.e. form a physical and/or a MAC layer link that may be either point-to-point or broadcast) must implement at least one common subclass of UseCoordination.
- XG systems that do not wish to form associations with each other at Layers 2 or higher, and merely wish to deconflict selected opportunities can also implement a common subclass of UseCoordination. For example, a cordless phone and a wireless data network interface may share a UseCoordination behavior even though they do not share a MAC layer and may not communicate (we mean data, not XG control signaling) with each other. A taxonomy of UseCoordination protocols is shown in Figure 63.

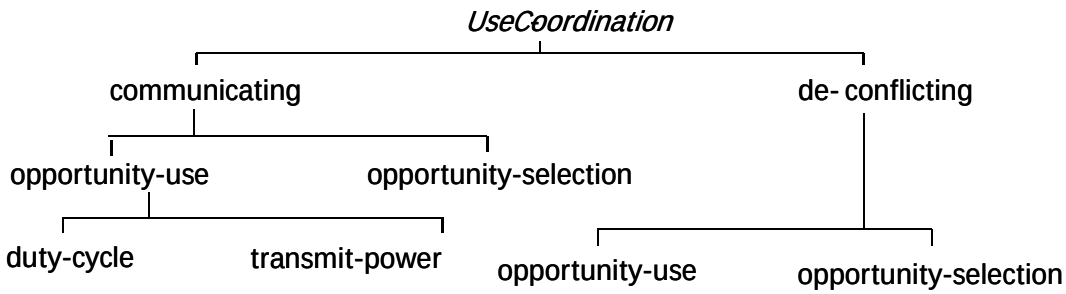


Figure 63: UseCoordination taxonomy

5 XG Information Objects

In this section, we describe the following XG information objects:

- **PolicyDefinedJoinPoint, Parameter, and Process classes**
- **Awareness, SensedAwareness, and ProtocolBasedAwareness classes**
- **OpportunityInstance and ValidOpportunityInstance classes**

Our goal in this RFC is not to specify any particular design of these classes, but rather to identify abstractions that are common to XG systems. We describe details of these classes only to the extent that is required in order to develop the abstract interfaces and behaviors in later sections. XG instantiations can extend these classes, and add additional classes as needed.

Other classes identified in Section 2.4 (including the ProtocolDataUnit class and its subclasses, the Expression class and its subclasses, the Credential class, the UsageRequest class, the ParameterCollection class, and the OpportunityInstance class) are open to implementation. We develop these classes further and provide example instances in Section 0

5.1 PolicyDefinedJoinPoint, Parameter, and Process

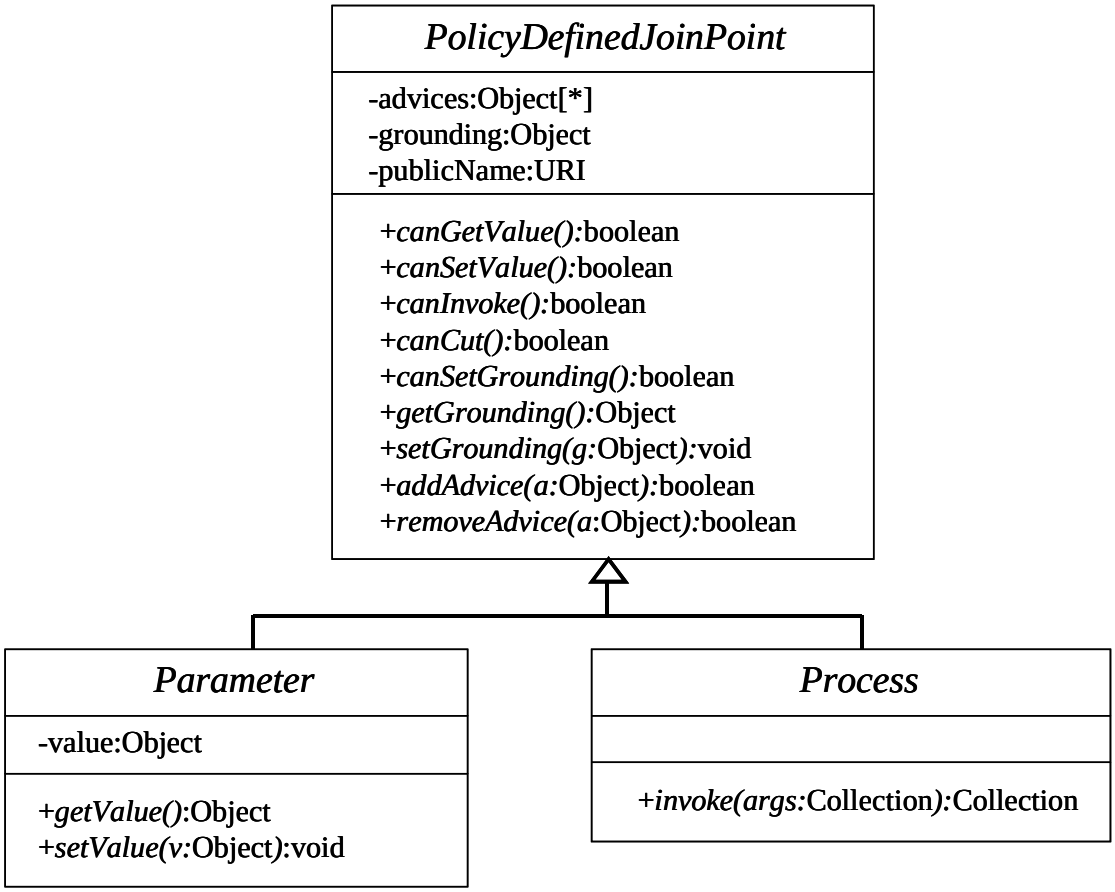


Figure 64: The PolicyDefinedJoinPoint class and its subclasses

The AccreditableKernel implements the SystemCapabilitiesInterface, through which access to all primitives (parameters and methods) implemented by the radio system is provided to the system strategy reasoner and the policy conformance reasoner. For this purpose, we propose an information object abstraction called PolicyDefinedJoinPoint that generalizes the Parameter and Process classes described in the XG Policy Language Framework RFC [XGPLF].

The PolicyDefinedJoinPoint class has a publicName attribute that takes a value of type Universal Resource Identifier (URI) corresponding to the primitive within the ontology that describes the capability of the XG radio system. Each such primitive must be grounded by an implementation within the radio, and a reference to this implementation is stored in the grounding attribute. It is important to note here that the actual implementation of the primitive may reside within any part of the system (e.g., sensor, transceiver, medium access control), but if the primitive is governed by regulatory or system policy (and specified externally through an ontology), then it must be accessible through the SystemCapabilitiesInterface.

The Parameter class extends the PolicyDefinedJoinPoint class by providing the *getValue* and *setValue* methods to access the value of radio parameters. The Process class extends the PolicyDefinedJoinPoint class by providing an *invoke* method in order to access a particular function implemented within the radio.

We envision that extensible XG radio systems will provide the flexibility to modify certain behaviors that are embedded within the system, for example, to take advantage of new opportunities enabled by policy. Modification of behaviors embedded within the system will require this ability to pre-empt (or *cut*) access to parameters and method invocations within the system, and then augmenting the program with additional logic. A unique feature, therefore, of the PolicyDefinedJoinPoint class (inspired by Aspect-Oriented Programming) is that it allows *cuts – advice* code can be inserted before, after, or around parameter accesses and process invocations. Two methods, *addAdvice* and *removeAdvice* are supported for this purpose. The actual implementation of the grounding and advice mechanisms are left to particular system design.

In addition the PolicyDefinedJoinPoint class provides methods that return a boolean value depending on whether the grounding of the primitive can be altered, and whether the implementation of the primitive supports access or modification of values, invocation of functions, or cuts where advice code can be inserted.

5.2 Awareness, SensedAwareness, and ProtocolBasedAwareness

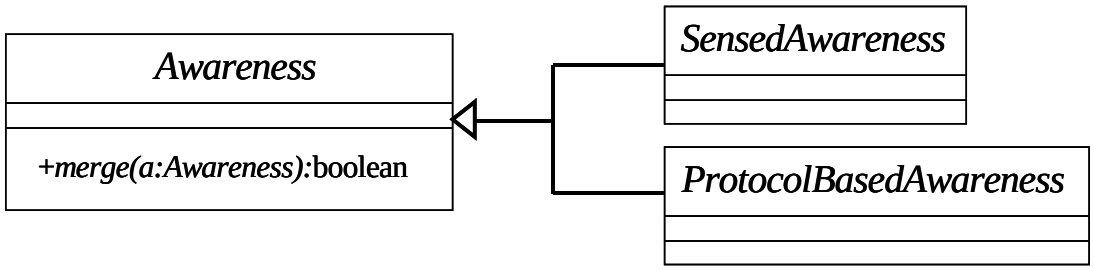


Figure 65: Awareness and its subclasses

The Awareness class encapsulates information that is acquired (or disseminated) by a radio regarding its spectral environment. Instances of the Awareness class must implement the *merge* method that enables information contained in one Awareness instance to be combined with another instance. This information consists of *spectral profile elements*, where each *spectral profile element* is a relationship between four entities: (i) a frequency specification, (ii) a time specification, (iii) a location specification, and (iv) a signal specification. For example, a spectral profile element can express the following relation: in the frequency interval 788-794 MHz (UHF TV channel number 67) during the time interval between 9:55:55.456789 and 9:56:55.437692 EDT on August 26, 2004, at latitude 42.3583, longitude 71.0603, and altitude 2m, observed a NTSC TV signal with 0.14mV/m. An Awareness instance could contain several such relations that span a wide range of frequencies. A concrete implementation is described in Section 0.

In order for the XG system to compose a composite awareness picture, each spectral profile element must be tagged with additional parameters that correspond to the source of the information, such as a sensor, or another node that provided the information.

The SensedAwareness class extends Awareness class with methods that provide access to the parameters governing the sensor and the sensing antennae. For example, such information can include the passband, integration time, and the frequency resolution settings of the sensor. The encapsulated sensed information can consist of raw samples in the time or frequency domain. Furthermore, sensed information can be energy-based and feature-based. Energy-based sensed awareness involves spectral characterization that does not take into account selectivity for specific signal features. For example, the power spectral density observed within a frequency spectrum can be reported. Feature characterization, on the other hand, takes into account specific features of the signal, emitter, or protocol.

The ProtocolBasedAwareness class extends the Awareness class with methods that provide access to parameters related to the identity of the nodes (that provided the spectral profile elements), and additional information provided those nodes regarding their capabilities and preferences for spectrum use. For example, a neighboring node could provide information regarding an idle channel, the spectrum that the neighbor will be listening to, when not actively transmitting to or receiving from other nodes. As

another example, in a secondary spectrum market context, a spectrum broker can include information about which frequency, time, or code slots, that the broker is willing to sell, or a node could present an auction bid for a particular frequency, time, or code slot. Protocol based awareness is acquired from other systems by interaction with the AwarenessDissemination behavior. Any environmental information acquired from other nodes is classified as disseminated even if the source of the information acquired it via sensing.

5.3 OpportunityInstance and ValidOpportunityInstance

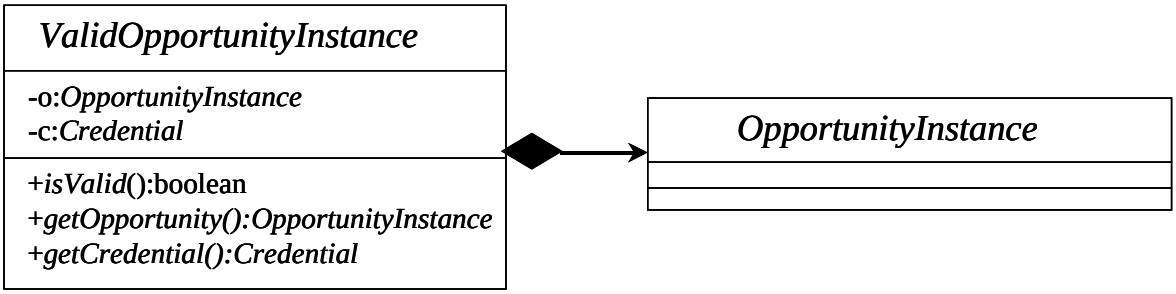


Figure 66: OpportunityInstance and ValidOpportunityInstance classes

The OpportunityInstance class must contain parameters that govern the authorized use of spectrum. These include parameters for sensor, transceiver, medium access control, awareness dissemination, use coordination, and other functions that are applicable to the particular XG system. No parameters or methods are defined for this abstract class, but a concrete implementation is described in Section 0.

ValidOpportunityInstance objects are returned by the *validateOpportunity* method of the PolicyInterface. A ValidOpportunityInstance class contains an OpportunityInstance class and provides three methods:

- The *isValid* method that returns whether the contained parameters conform to policy
- The *getCredential* method that returns a Credential object generated by the policy conformance reasoner
- The getOpportunity method that returns the contained OpportunityInstance object

6 Reference System Design Based on XG Abstract Behaviors

In this section, we describe a design sketch of a concrete XG radio system based on the abstractions described in the earlier sections. This particular design is only intended as an example to illustrate one way in which some of the key abstractions can be realized. Numerous other approaches to the design of XG systems than the one presented here are indeed possible.

The reference radio system extends and concretely implements the *XgRadioSystem* described in Section 4.1.1. The reference radio system contains implementations of concrete classes that extend each of the subsystems, behaviors, interfaces, and

information objects described earlier. A complete specification of each of these concrete classes is beyond the scope of this document.

Rather, our goal in this section is to illustrate how multiple instances of the reference radio system can establish and maintain data communications within an opportunistic spectrum-sharing environment under policy constraints. For this purpose, we restrict our focus to the XG Behaviors within the *AccreditableKernel* subsystem, and their interactions with the *MediumAccessControl* subsystem. As an example, we describe a protocol based on the CSMA family of protocols that we have extended to operate within the XG environment.

We introduce concrete classes that extend each of the four XG Behaviors described in Section 0, and are contained within the *AccreditableKernel*, and describe how these XG Behaviors interact with various XG Interfaces described in Section 0. As part of the description, we will introduce as needed concrete classes that extend three XG Information Objects described in Section 0.

In this section, we describe the design and operation of the reference radio system in five steps:

1. **Acquisition of spectrum awareness through sensing:** We introduce the HoleVector that extends *Awareness,* which is central to the acquisition, management, and dissemination of spectral awareness information, and the HoleInformationManagement class that concretely implements the *SpectrumAwarenessManagement* behavior. We also introduce a second class, SensorUpdate, which extends *SensedAwareness*.
2. **Topology management and awareness dissemination through a protocol:** We introduce the NeighborTable class that extends *ProtocolBasedAwareness.* We describe the NeighborDiscoveryAndHoleInformationProtocol that extends *AwarenessDissemination;* information is exchanged between radio systems using this protocol through instances of the NeighborPacket class that extends *ProtocolDataUnit*. The NeighborPacket contains a NeighborUpdate, which may be compressed, encrypted, and may carry integrity and authenticity checks.
3. **Opportunity allocation and use under policy constraints:** We describe the HoleUseManagement class that concretely implements the *UsageAccountingManagement* behavior. We introduce the ChannelRequest class that extends *UsageRequest* as well as the AllocatedChannel class that extends *OpportunityInstance.*
4. **Coordination of use through idle channel selection:** We present the IdleChannelSelection that concretely implements the *UseCoordination* behavior. In our approach, this behavior works in close coordination with the medium access control.
5. **Medium access control within an opportunistic spectrum-sharing environment:** We present the XMediumAccessControl that concretely implements the *MediumAccessControl* subsystem. XMediumAccessControl works closely with the behaviors implemented within the *AccreditableKernel* of the radio system, in particular, the HoleUseManagement and IdleChannelSelection behaviors.

6.1 Acquisition of Spectrum Awareness Through Sensing

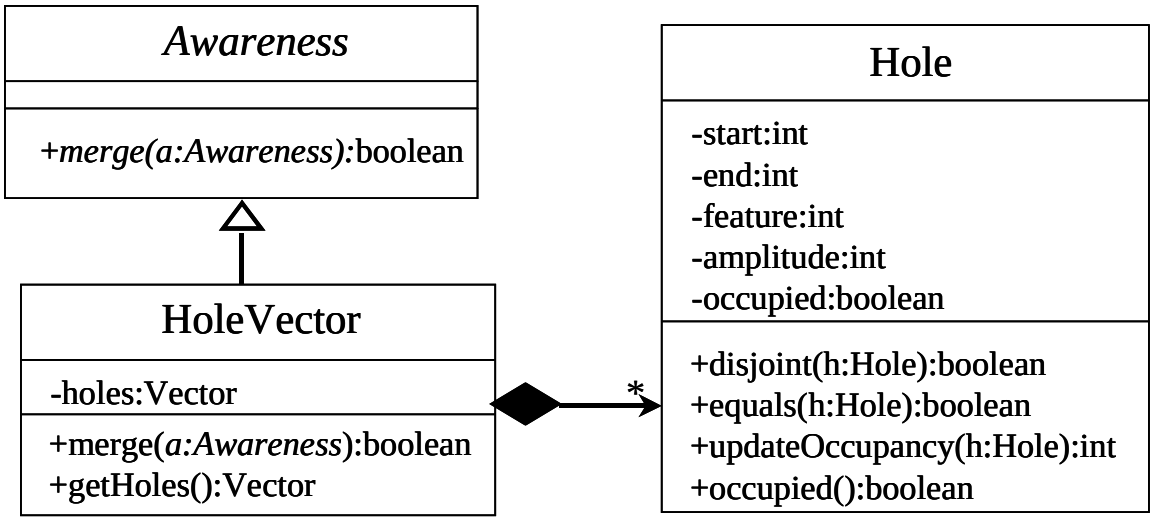


Figure 67: HoleVector is a concrete class extending Awareness

We introduce the Hole class that stores information about a frequency interval. The Hole class has start and end fields to store the limits of the interval, a feature field that stores information about the feature being detected within that interval, such as an incumbent signal type, an amplitude field to store information such as the received signal strength or power spectral density corresponding to the feature, and an occupied field to indicate that this interval is unavailable for use by the XG system.

The Hole class implements a disjoint method to determine if another instance provided as an argument has any overlap with the given instance, an equals method to determine if another instance equals the given instance, and an updateOccupancy method which updates the occupancy field based on another instance provided. For illustration, we

```
boolean disjoint(Hole h) {
  return ((start > h.end) || (end < h.start));
}

boolean equals(Hole h) {
  return ((start == h.start) && (end == h.end)
          && (feature == h.feature));
}

int updateOccupancy(Hole h) {
  if(disjoint(h)) { return 0; }
  if((occupied == true) || (h.occupied == false)) { return 1; }
  occupied = true;
  return 2;
```

provide pseudo-code (in Java-like syntax) for these methods.

Instances of the Hole class are contained within a HoleVector, a class that extends *Awareness*. The HoleVector class implements a getHoles method that returns the Vector containing instances of the Hole class. HoleVector also implements a merge method

(illustrated in the pseudo-code below) that processes a Vector of Hole instances provided. For each element in the Vector, the updateOccupancy method is called. New information (if any) is added to this instance, and the method returns a boolean value indicating whether the HoleVector was changed as a result of the merge. The HoleInformationManagement class calls this method whenever new awareness is received from either
the sensor or through a dissemination protocol.

```
boolean merge(Awareness a) {
  boolean sc = false;
  for(Hole i : a.getHoles()) {
    boolean newinfo = true;
    for(Hole j : holes) {
      int result = j.updateOccupancy(i);
      if (result == 2) { sc = true; }
      if (result > 0) { newinfo = false; }
    }
    if (newinfo == true) { holes.add(i); }
  }
  return sc;
}
```

HoleInformationManagement concretely implements the SpectrumAwarenessManagement behavior. This class has a myAwareness field that is initialized by the constructor to hold an instance of HoleVector. This instance is returned by the getAwareness method. The constructor also initializes references to a SensorInterface and an AwarenessDisseminationServiceAccessPoint. The HoleInformationManagement class has another field called idleChannel that stores a Vector of Hole instances. The purpose of this field is to determine a set of opportunities to which the XG system is listening when it is not actively transmitting or receiving so that other systems may contact it. The IdleChannelSelection class described later does the computation of this field.

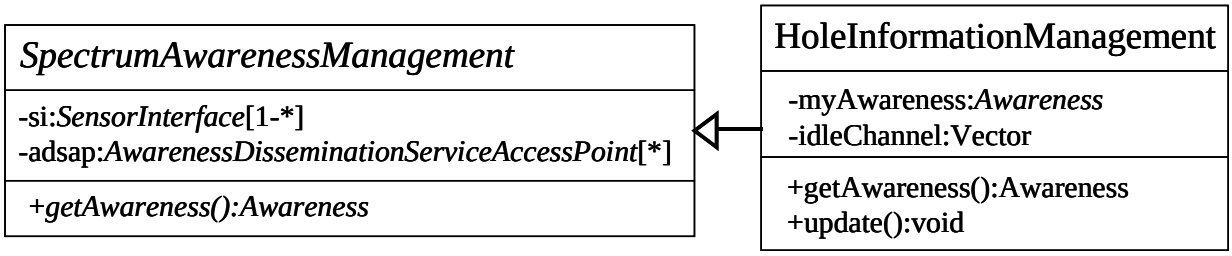


Figure 68: HoleInformationManagement extends SpectrumAwarenessManagement

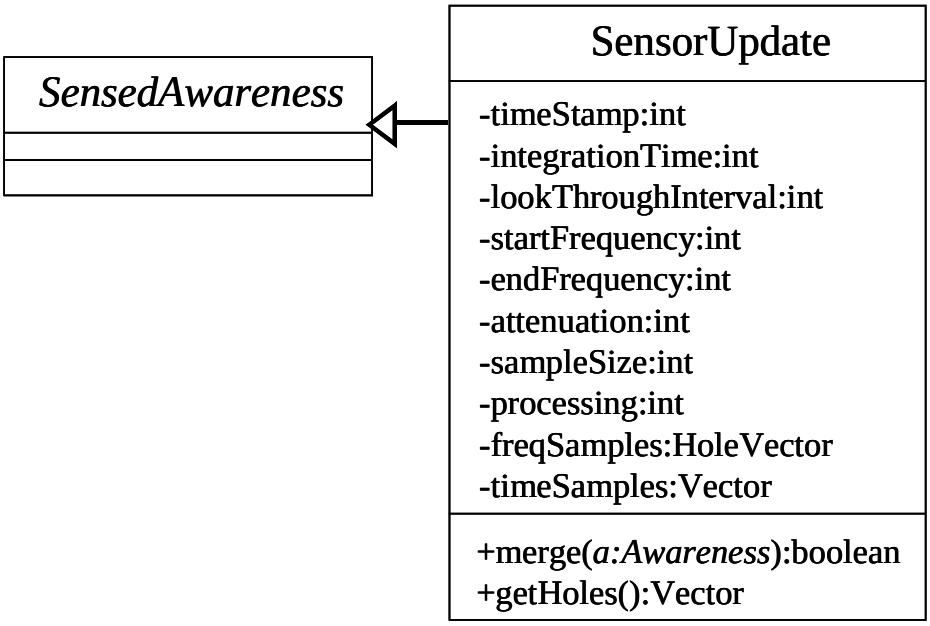


Figure 69: SensorUpdate a concrete class extending SensedAwareness

The update method of HoleInformationManagement behavior (see pseudo-code below) implements the key operations of this behavior, which consists of (i) merging the awareness information received through sensing or through a protocol when notified, and (ii) recomputing the idle channel as well as disseminating the awareness if there is a change as a result of the merging.

The SensorUpdate class we present below is an example of what a sensor may provide either periodically or in an event-driven fashion. The SensorUpdate class has a freqSamples field to hold frequency samples and a timeSamples field to optionally carry

```
void update(Interface i) {
  boolean sc = false;
  if (i instanceof SensorInterface) {
    sc = myAwareness.merge(i.getSensedAwareness());
  } else if (i instanceof AwarenessDisseminationServiceAccessPoint) {
    sc = myAwareness.merge(i.getProtocolBasedAwareness());
  }
  if (sc) {
    idleChannel = IdleChannelSelection.recomputeIdleChannel(myAwareness);
    adsap[0].disseminate(myAwareness);
  }
}
```

raw time samples. Additional fields store related sensor parameters such as the time stamp, the sample size, the type of processing (such as FFT using a Hanning window, or ATSC TV signal detection), the time interval between successive samples, the integration time for the sensor, the start and end frequency of the band covered by the sensor, and the attenuation setting.

The SensorUpdate class has a merge method that may be called by the sensor for example to perform smoothing and averaging of the sample prior to presenting it to HoleInformationManagement.

We describe a class to encapsulate protocol based awareness and a concrete subclass of the AwarenessDissemination behavior next.

6.2 Topology Management and Awareness Dissemination

The NeighborTable class extends and concretely implements *ProtocolBasedAwareness.* Instances of NeighborTable contain instances of the NeighborUpdate class. Each instance of NeighborUpdate encapsulates topology and sensed spectral information from a single XG system. The NeighborUpdate includes the neighbor's identifier, a time stamp, the transmit power used by the neighbor so that path loss information can be computed, the processing gain, the signal type used by the neighbor to communicate, and the idle channel where the neighbor will listen when it is not actively transmitting or receiving. NeighborUpdate optionally includes topology information as a vector of Adjacency instances containing the neighbors of the node sending the update, whether those

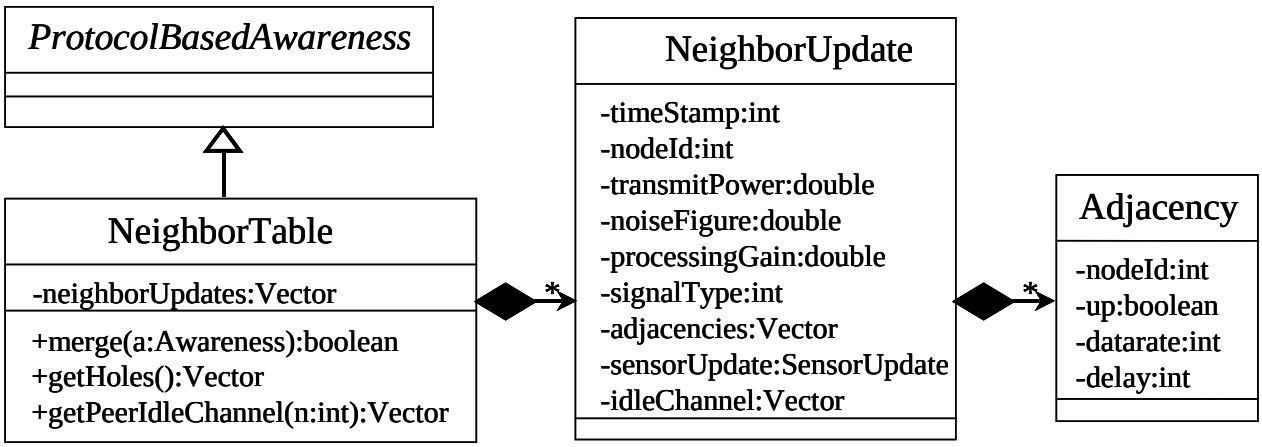


Figure 70: Classes for protocol-based awareness

neighbors are up or down, and additional parameters such as datarate and delay. NeighborUpdate also includes a SensorUpdate instance based on its sensed awareness. Information is exchanged between radio systems using instances of the NeighborPacket class. The NeighborPacket class extends *ProtocolDataUnit* contains a NeighborUpdate, which may be compressed, encrypted, integrity and authenticity checks. NeighborPacket includes typical protocol header information such as addresses, sequence numbers and time to live fields.

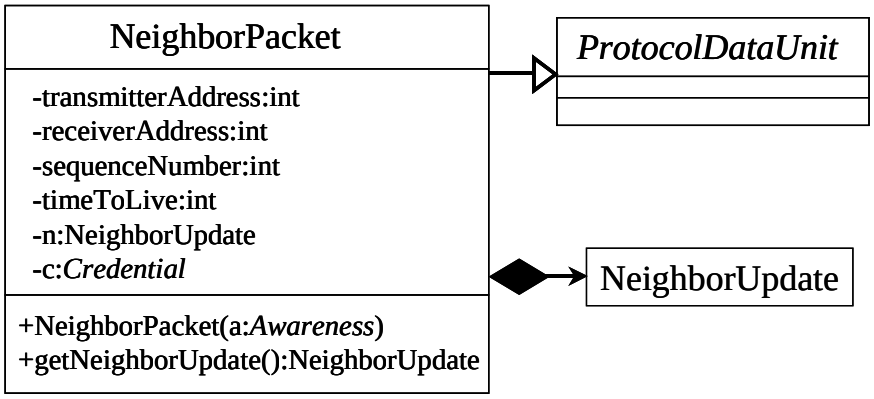


Figure 72: NeighborPacket extends ProtocolDataUnit for Awareness Dissemination

The NeighborDiscoveryAndHoleInformationProtocol class concretely implements the AwarenessDissemination behavior. The constructor of this class initializes the vcc field with an instance of a VirtualCoordinationChannelInterface to be used for dissemination, and a neighborTable field that stores protocol based awareness. This behavior also

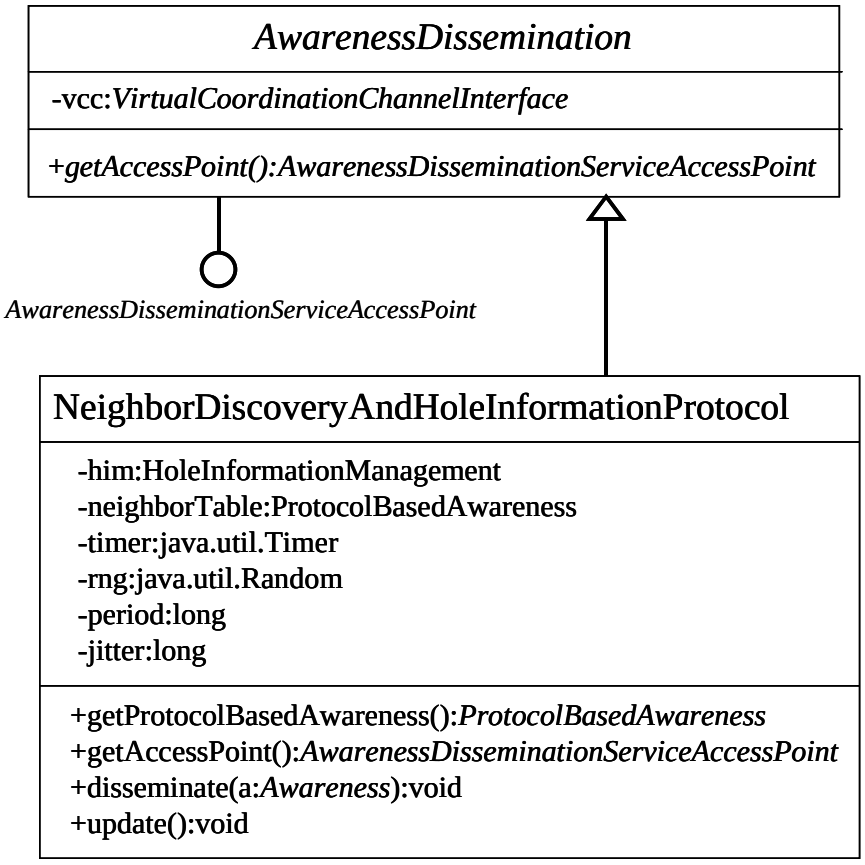


Figure 71:NeighborDiscoveryAndHoleInformationProtocol behavior

includes a timer field and a random number generator that are also initialized by the constructor.

The key functionality of this class is implemented by the update method and the disseminate method. Pseudo-code for these methods is provided below. The virtual coordination channel calls the update method whenever a new NeighborPacket is received, causing the contained NeighborUpdate to be merged with the NeighborTable. If there is a change as a result of the merge registered observers (e.g. HoleInformationManagement) are notified.

The disseminate method is called either by HoleInformationManagement when there is a change as a result of a merge, or periodically when the timer fires. When called, this method cancels any pending timer request, constructs and sends a NeighborPacket through the VirtualCoordinationChannelInterface, and sets a timer again to fire after a time with a small random jitter added. In order to construct the NeighborPacket the

idleChannel and the SensorInterface fields of the HoleInformationManagement instance may be accessed.

A utility class Dissemination has been defined to define a run method to be executed when the timer fires. The run method calls the disseminate method.

```
class Dissemination extends java.util.TimerTask() {
  public void run() { disseminate(mp); }
}

void update(Interface i) {
  boolean sc = false;
  if (i instanceof ServiceAccessPoint) {
    Object p = i.receive();
    if (p instanceof NeighborPacket) {
      sc = neighborTable.merge(p.getNeighborUpdate());
    }
  }
  if (sc) { notifyObservers(); }
}

void disseminate(Awareness a) {
  timer.cancel();
  vcc.send(NeighborPacket(a));
  timer.schedule(new Dissemination(), period + rng.nextLong(jitter));
```

6.3 Opportunity Allocation and Use under Policy Constraints

We describe the HoleUseManagement class that concretely implements the *UsageAccountingManagement* behavior. We note that the allocation functionality is embedded within HoleUseManagement and this class implements the AllocationInterface.

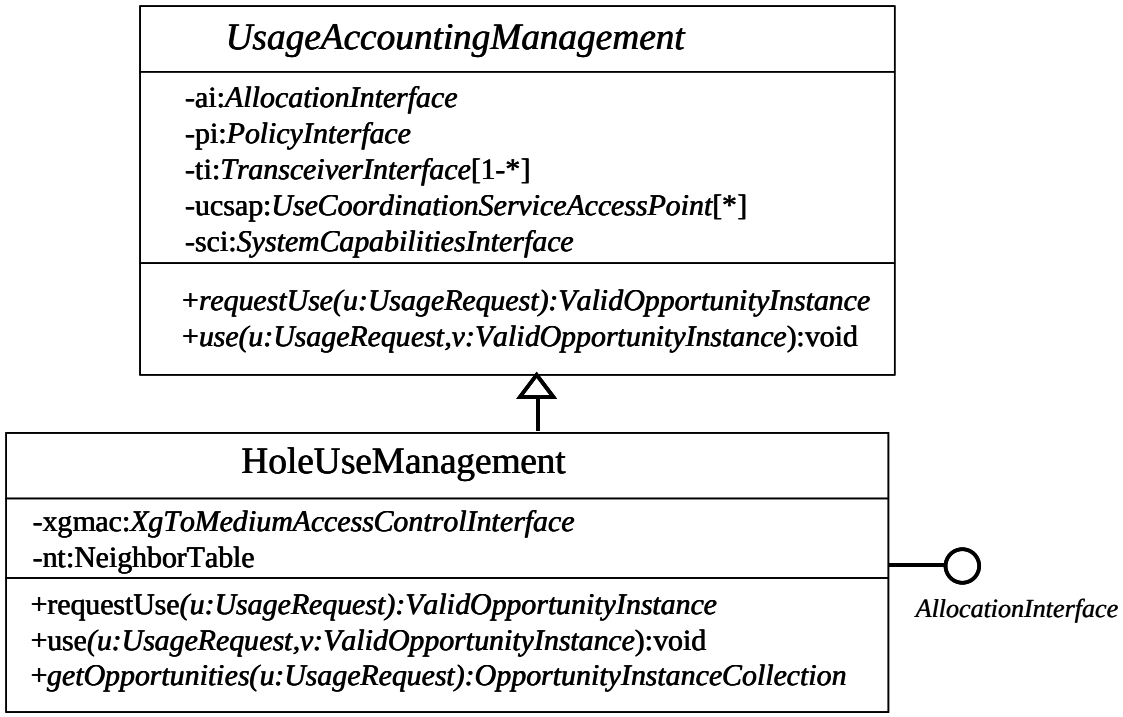


Figure 72: HoleUseManagement implements UsageAccountingManagement and AllocationInterface

The ChannelRequest class extends the *UsageRequest* class to enable concrete requests for opportunities using the requestUse method. The ChannelRequest class contains fields to specify whether an opportunity to transmit or receive data is requested, and the type and length of the frame to be transmitted. Optionally the peer to whom communications is intended, the particular channels requested, and the transmit power spectral density can also be specified.

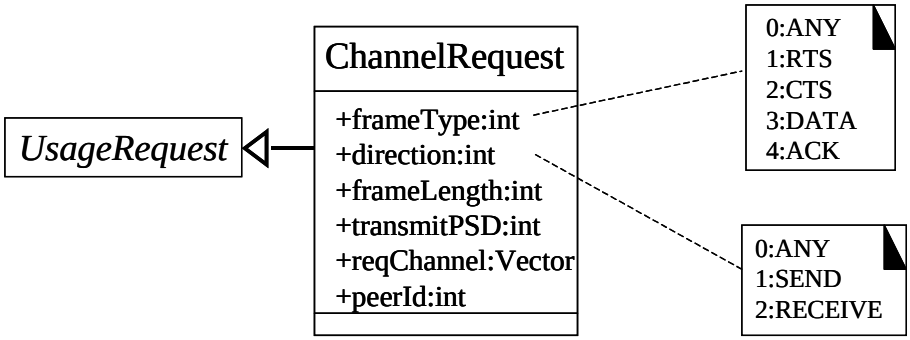


Figure 73:ChannelRequest extends the UsageRequest class

ChannelRequest and the HoleUseManagement behavior are tailored for use with XMediumAccessControl that we describe in Section 0.

When the requestUse method is invoked with a ChannelRequest for receiving ANY frame (including an RTS) from any unspecified neighbor, the computed idle channel for this radio is returned in an AllocatedChannel instance. When the requestUse method is invoked with a ChannelRequest for transmitting an RTS to a peer, the peer's idle channel as well as a channel for communicating data with the peer are calculated from the neighbor table and returned. In the case of sending a CTS or ACK to a peer, the peer's RTS would contain the channels requested for transmission. These channels must be validated as being unoccupied against the sensed and protocol based awareness prior to use. Prior to returning an AllocatedChannel within a ValidOpportunityInstance, it is always validated by submitting it to the PolicyInterface to ensure that the opportunity (described by the channels and other parameters such as the transmit power spectral density) is authorized by policy.

When the use method of HoleUseManagement is invoked, the channels to be used are validated against the sensed and protocol based awareness to ensure they are unoccupied. Then the transceiver is notified of the channels to tune into by calling the use method of TransceiverInterface.

In order to concretely describe opportunities that are allocated and authorized, we introduce the AllocatedChannel class that extends *OpportunityInstance.*

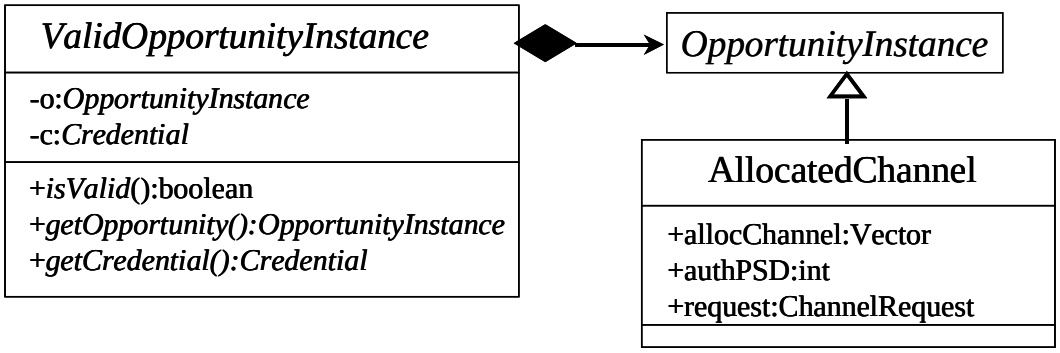


Figure 74: AllocatedChannel extends OpportunityInstance

6.4 Use Coordination Through the Selection of an Idle Channel

We present the IdleChannelSelection that concretely implements the *UseCoordination* behavior.

In our example, coordination between communicating nodes is done by the use of an idle channel. Each node tunes to, and listens on, the idle channel whenever the node is not actively transmitting or receiving data packets. Furthermore, each node conveys its idle channel to its neighbors using the NeighborDiscoveryAndHoleInformationProtocol;

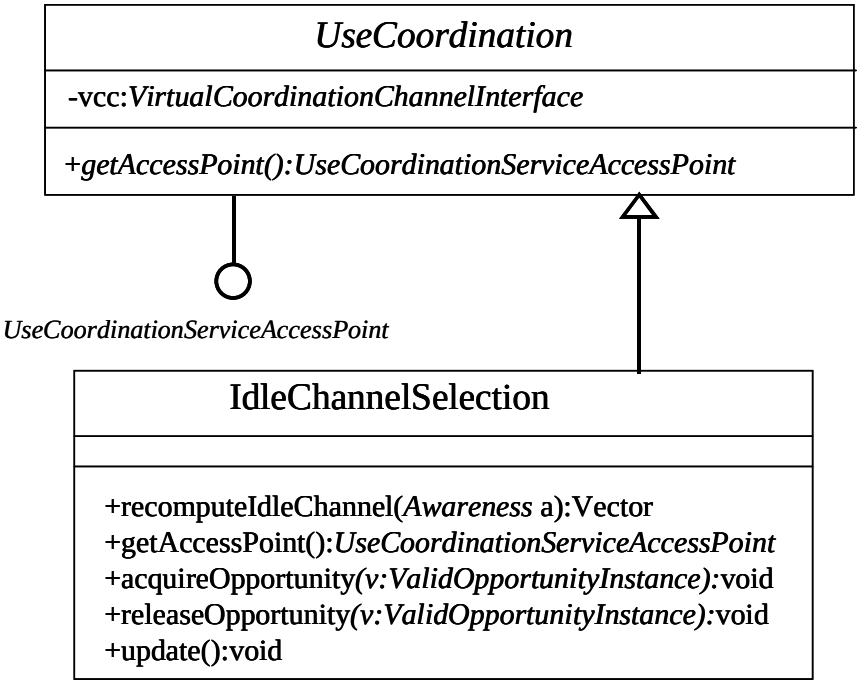


Figure 75: IdleChannelSelection behavior

therefore, neighbors can contact the node whenever it is available.

The IdleChannelSelection class implements the recomputeIdleChannel method, which computes and returns the largest contiguous set of unoccupied Hole instances contained

in the HoleVector that is passed to the method as an argument. As discussed in Section 0, the recomputeIdleChannel method is invoked with the merged awareness, therefore, the computed idle channel will include only those holes that are available to all neighbors. When new spectrum awareness is acquired through sensing or a dissemination protocol, the update method of HoleInformationManagement invokes the recomputeIdleChannel method with a HoleVector instance as argument. This HoleVector instance contains merged occupancy information both from sensing locally and information provided by neighbors. Effectively, this results in the node choosing an idle channel that is the largest contiguous spectrum available to all of its neighbors.

In our example XG system, the signaling for coordination between neighbors to establish communications is done by the medium access control as described in Section 26.5. The update, acquireOpportunity and releaseOpportunity methods of IdleChannelSelection are currently empty. They may however be modified to invoke any coordination procedures that are required by policy. In this case, the names of these coordination procedures and related parameters will be contained within the validated opportunity instance. For example, the reuse of public safety bands based on the reception of an authorization beacon, the acquireOpportunity procedure must receive and decode the authorization beacon.

6.5 Medium Access Control for Opportunistic Spectrum-Sharing

In this section, we present XMediumAccessControl that concretely implements the *MediumAccessControl* subsystem. XMediumAccessControl (XMAC) works closely with the behaviors implemented within the *AccreditableKernel* of the radio system, in particular, the HoleUseManagement and IdleChannelSelection behaviors. XMAC is derived from the CSMA family of protocols and represents one of many possible approaches.

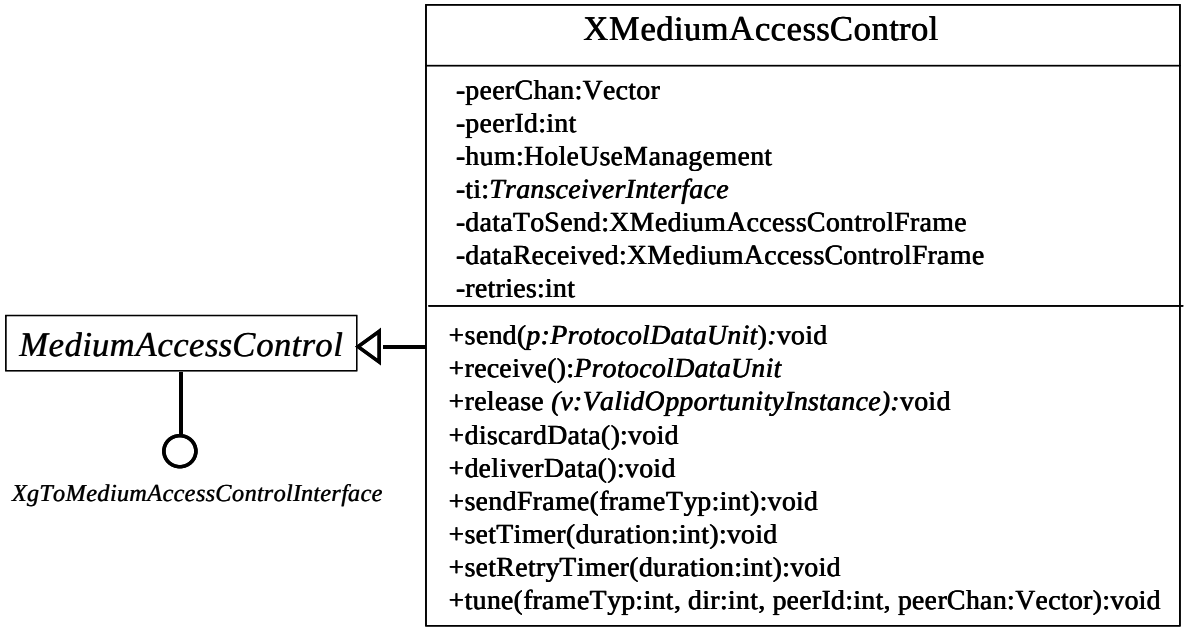


Figure 76: XMediumAccessControl extends MediumAccessControl

Next we define the XMediumAccessControlFrame class that extends the ProtocolDataUnit for use with XMediumAccessControl.

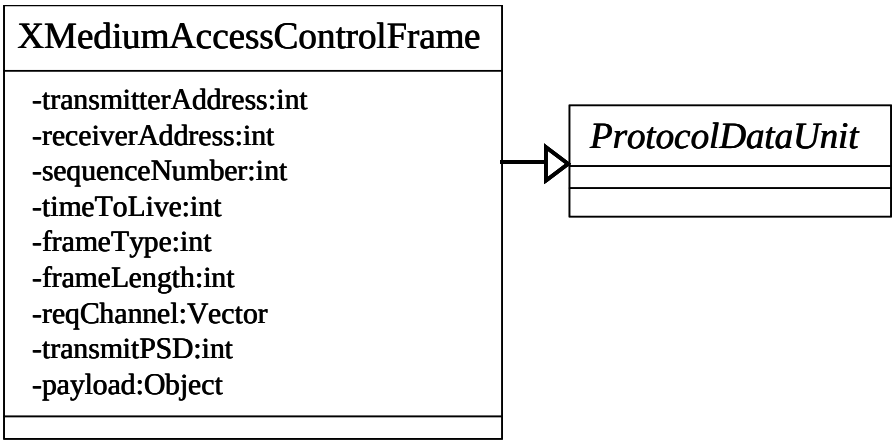


Figure 77: XMediumAccessControlFrame is a ProtocolDataUnit

The frameType field can take four different values: RTS, CTS, Data and ACK. Before XMediumAccessControlFrame instances can be sent are received the XMediumAccessControl must construct corresponding ChannelRequest objects and invoke the requestUse method of HoleUseManagement to obtain AllocatedChannel instances.

We provide a state transition diagram for XMAC in Figure 77. Some details (such as handling errored frames and other error conditions including release of opportunities e.g., when idle channel is recomputed) are not shown here for clarity.

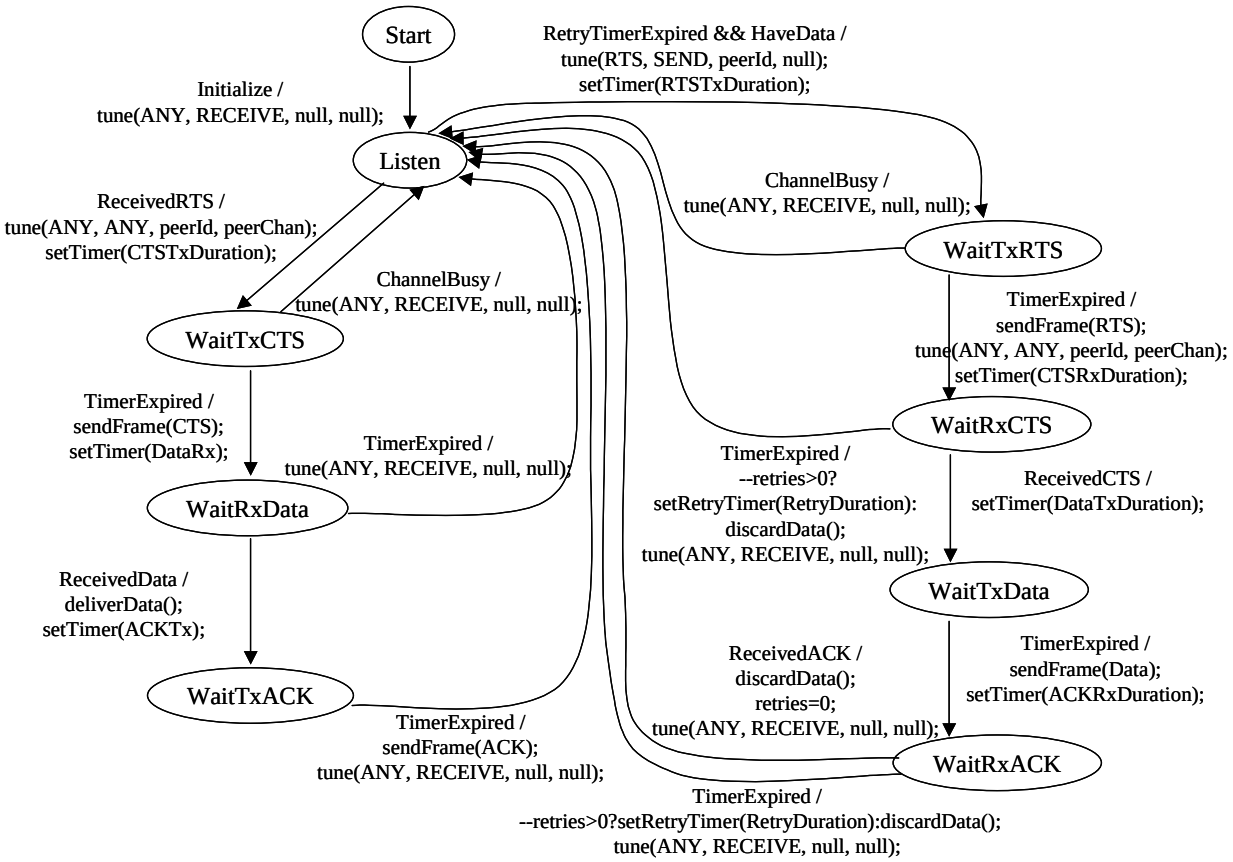


Figure 78: State Transition Diagram for XMediumAccessControl

Upon initialization, the tune method is invoked to receive ANY frame from any neighbor. This in turn calls the requestUse method of HoleUseManagament, which allocates the idle channel of the node, validates the idle channel against policy and returns a ValidOpportunityInstance. Upon receiving this, the tune method calls the use method of HoleUseManagement, which instructs the transceiver to tune to and receive the idle channel after asserting the idle channel is not occupied by the node and its peers. Then XMAC enters the Listen state. Whenever the node is not actively transmitting or receiving, this step is repeated and the transceiver is tuned to the idle channel, and XMAC returns to the Listen state.

The tune method manages the interaction between XMediumAccessControl and HoleUseManagement and therefore plays a critical role in the operation of this XG implementation.

The left hand portion of the state diagram contains states involved in the reception of data. Upon receiving an RTS from a neighbor (peerId), a UsageRequest containing the requested channels for communication contained in the RTS (peerChan) is sent to HoleUseManagement. If these channels are allocated and validated, then the transceiver is instructed to tune to these channels. Then XMAC sets a timer by calling setTimer(CTSTxDuration) and enters the WaitTxCTS state. In this state, the node must continue to sense the channel for other communications (e.g. incumbent signals) in its

vicinity, and if any such activity is detected XMAC returns to the Listen state. This time also ensures that a sufficient interframe spacing is achieved to account for propagation delays within the network. If no other activity is detected until the timer fires, XMAC sends a CTS message to peerId using peerChan, sets a timer using setTimer(DataRxDuration), and enters the WaitRxData state. Upon receiving the CTS, the peer will send a Data packet. If a Data frame is not received before the timer fires, XMAC returns to the Listen state. When a Data frame is received, XMAC sets a timer (for interframe spacing) by calling setTimer(ACKTxDuration), delivers the data to the higher layers by calling deliverData() and enters the WaitTxACK state. When the timer fires, XMAC sends an ACK frame to the peer and returns to the Listen state.

The right hand portion of the state diagram contains states involved in the transmission of data. Whenever a higher layer provides a frame to be transmitted to another node (peerId), the data frame is stored in the dataToSend field, and the retries variable is set to the maximum retry count (a configurable positive integer). A retry timer is set to fire immediately by calling setRetryTimer(0) – the event of receiving a data frame from the higher layer is not shown in the state diagram. When XMAC is in the Listen state, if a data frame is waiting to be sent and the retry timer has expired, then a UsageRequest is made to HoleUseManagement for sending an RTS to peerId. This request results in both the idle channel of the peer and also allocated another (possibly larger or less crowded) set of channels for communicating with this peer (peerChan). The returned peerChan will be included in the RTS frame.

The transceiver is instructed to tune to the idle channel of the neighbor (peerId) to whom the data frame must be sent. XMAC sets a timer by calling setTimer(RTSTxDuration) and enters the WaitTxRTS state. If the channel is never free until the timer fires, XMAC decrements the retries variable and returns to the Listen state. If the retries variable is zero, then the frame is discarded. If the channel is free, XMAC sends an RTS to peerId with the requested channel for the communication (peerChan). After the RTS has been sent, the transceiver is instructed to tune to peerChan. XMAC then sets a timer by calling setTimer(CTSRxDuration), and enters the WaitRxCTS state. If a timeout occurs, it increments the retries and returns to the Listen state. If a CTS frame is received, then XMAC sets a timer by calling setTimer(DataTxDuration) to enforce an interframe spacing, and enters the WaitTxData state. When the timer fires, XMAC sends the Data frame to the peer, sets a timer by calling setTimer(ACKRxDuration) and enters the WaitRxACK state. If an ACK frame is received before the timeout occurs, the data has been successfully transmitted and acknowledged, and the stored copy is discarded. Otherwise, the retries field is decremented. If additional retries are permitted the retry timer is set, otherwise, the data frame is discarded. In both cases, XMAC returns to the Listen state.

The description above is intended for illustration only and several optimizations are possible. For example, an obvious enhancement is to avoid the use of RTS/CTS exchange by directly transmitting in the idle channel of the peer in the case of small frames.

7 Summary and Future Work

This document defines an abstract XG radio system in terms of three kinds of abstractions:

- Interfaces
- Behaviors
- Information Objects

We have defined nine interface classes, and four behavior classes (two internal behaviors, and two protocol behaviors) for XG systems. In addition, we have developed three classes of abstractions for XG information objects – PolicyDefinedJoinPoint, Expression, and ProtocolDataUnit classes. We described in detail the key information object abstractions including the PolicyDefinedJoinPoint, the Awareness, and the OpportunityInstance classes. Taken together, these interfaces, behaviors, and information objects succinctly capture the characteristics envisioned of XG systems [XGV] – opportunistic spectrum sharing, policy-defined operation, and traceability of physical behaviors to a defined set of abstract behaviors.

This RFC is focused on full-featured XG systems. In future work, we envision that specific XG system implementation profiles will be added, for example, to support decentralized spectrum broker architectures, or to support low-cost XG end systems which rely on other XG systems for opportunity awareness, allocation, and use.

Formal description of radio instances is necessary in order to convey the goals, capabilities, and operational constraints of the radio to the system strategy reasoner. In future work, we envision the development of a formal description framework that will build upon the XG policy language framework.

In the interest of achieving basic interoperability across all XG systems (to the extent necessary to bootstrap communications between disparate XG systems), we recommend that the community adopt at least one concrete subclass of the AwarenessDissemination and UseCoordination behaviors to be implemented by all XG systems. The specific protocols to be standardized for this purpose are left for future work. Formal specification and verification of the abstract behaviors and the protocols (e.g., using formalisms such as Timed Automata) is also the subject of future work.

Future versions of this document may also include another section that describes in detail a use-case for the XG implementation described in Section 0. This use case should include particular network scenarios with incumbents and other XG nodes, the needed spectrum awareness and spectrum use primitives along with an explanation of how the primitives may be implemented as PolicyDefiunedJoinPoint instances within a XG radio, policy assertions and inference rules, and assertions regarding sensed and protocol-based awareness along with rules to infer valid idle channels for a node and communication channels for a pair of nodes. The use case should also include examples of opportunity

search and validation and explain how search and validation are instances of theorem proving within the knowledge base of policy and awareness (incuding assertions regarding use that result from the search).

In order to better understand the abstract behaviors in the context of policy-driven operation, and in order to offer guidance to XG system designers, we plan to develop concrete instances of the behaviors.

We plan to validate and incrementally refine the abstract behaviors using experience gained from BBN's XG system simulation platform. Future versions of this document are expected to incorporate the results of these efforts.

We seek feedback on this document from the government, the XG community, and the public though the RFC process, to be incorporated in future versions as appropriate.

Acknowledgments

This document was prepared by the Internetwork Research Department, BBN Technologies, with input from DARPA/ATO and the DARPA XG Working Group.

Comments

Comments on this RFC should be emailed to the document editors at xg-rfc-comments@bbn.com, along with the commenter's name and organization.

Bibliography

[XGV] XG Vision RFC, http://www.darpa.mil/ato/programs/XG/rfc_vision.pdf
[XGAF] XG Architectural Framework RFC, http://www.darpa.mil/ato/programs/XG/rfc_af.pdf
[XGPLF] XG Policy Language Framework RFC, http://www.darpa.mil/ato/programs/XG/rfc_policylang.pdf
[UML] The Object Management Group, Unified Modeling Language, http://www.omg.org/technology/uml/
[DP] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object Oriented Software, ISBN: 0201633612, Addison-Wesley, 1995.

Appendix D
XG Policy Language Framework
Request for Comments

Version 1.0
April 16, 2004

1 Introduction

This section provides the context for this document. It consists of the following sections:

- ***Background:*** describes the XG program in general as well as XG reference documents and how they are intended to evolve and to be used in the XG program and beyond.
- ***Motivation:*** explains why machine understandable policies are required for spectrum agile radios.
- ***Requirements:*** summarizes the underlying properties deemed necessary for the XG Policy Language framework to meet its objectives in the XG program.
- ***Evolution:*** briefly describes how the policy language framework is designed to support future evolution.
- ***Document Roadmap:*** provides a guide to later sections of this document.

Definitions of certain terms and expansions of acronyms used in this document are provided in Appendix A. Bibliographic citations in the document are enclosed in square brackets (e.g. [XGV]) and a list of references is provided later (see Bibliography).

1.1 Background

We briefly summarize the premise and goals of the XG program and the documents associated with that program.

XG (neXt Generation Communications) is a DARPA-funded research program based on the (now generally accepted) premise that the historic (and current) method of authorizing spectrum use–static, administrative allocation–results in an apparent scarcity of spectrum that can be avoided by the proper application of dynamic spectrum sharing techniques. The goals of the XG program are:

1. Demonstrate through technological innovation the ability to utilize available (unused, as opposed to unallocated) spectrum more efficiently.
2. Develop the underlying architecture and framework required to enable the practical application of such technological advances.

As part of the second goal the XG program, participants (known collectively as the XG Working Group or XG-WG) will produce a series of Requests for Comments (RFC) that together describe the proposed XG architecture and framework. This is one of those documents. These documents are RFCs because the authors recognize that the final development of such an important technology cannot be accomplished by a small group of individuals. It requires input and participation from a broad representation of the affected community. Thus it is hoped these RFCs will spur that community to provide feedback in order to assure an organized and technically valid approach to the evolution of this architecture.

This RFC, XG Policy Language Framework, is the third to be released. The XG Policy Language Framework follows the XG Vision] and XG Architectural Framework]. The XG Vision RFC provides the motivation and scope of dynamic spectrum sharing envisioned by the XG program and describes an approach for developing the XG

architecture. It is highly recommended that the XG Vision RFC be read before (the rest of) the XG Policy Language Framework RFC. The XG Architectural Framework RFC presents the XG architecture, system components, and a high level concept of operations for XG communications. The XG Abstract Behaviors RFC will be released in the near future. It will provide an abstract view of the operation of XG radios, so that policies may be written to govern the operation of radios that are yet to be engineered.

1.2 Motivation for Machine-Understandable Policy

XG-enabled radios will be able to utilize available spectrum intelligently based on knowledge of actual conditions rather than using current conservative spectrum management methods (static spectrum assignments). In this way, XG technologies will utilize spectrum in a much more efficient manner than it is today. Furthermore, without the need to statically allocate spectrum for each use, networks can be deployed much more rapidly. Today the military, for example, must make spectrum use requests to their spectrum managers, who must deconflict them, and make static assignments far in advance of deployment. With XG capabilities, this process can be significantly shortened or perhaps even eliminated.

Radios must adhere to rules that apply to their operation. A major intent of such rules is to reduce or avoid interference among users. Such rules may cover both transmission and reception functions of a radio. Currently such rules are enumerated in spectrum policy as produced by various spectrum authorities such as the FCC or the NTIA in the U.S.A. In this document we use the term *spectrum policy* to refer to any externally (to the radio) imposed rules for spectrum use.

A radio that is capable of dynamically utilizing spectrum must be able to adhere to rules corresponding to the many uses of which it is capable–not just one use, as with most current radios. XG radios will be expected to operate over a wide range of frequencies and within different geopolitical regions. Therefore, they must incorporate a real-time adaptive mechanism for conforming to the policies applicable to each situation. In other words, XG radios must be *policy-agile*, by which we mean both that the radios are situationally adaptive to the current policy, and that they allow policy to be dynamically updated as well.

Spectrum policies relevant to a given radio may vary in several ways:

1. Policies may be altered over time.
2. The radio (along with its user) may move from one policy administrative domain to another (e.g. a military user may be deployed to a different country).
3. Policies may be dependent on time of day or year.
4. A spectrum owner/leaser may impose policies that are more stringent than those imposed by a regulatory authority.
5. The spectrum access privileges of the radio may change in response to a change in radio user.

To be truly versatile, the XG radios should be responsive to such changes in policy.

Current radios support only a small number of modes of operation and a limited range of intended operating environments. With only limited hardware agility, all the relevant policy sets that apply can be hard-coded into the radio. However, with the increasing agility and programmability of radio hardware (as illustrated in Figure 79), especially when combined with the prospect of opportunistic spectrum sharing, both the number of modes of operation as well as the range of operating environments for the radio will increase tremendously. As a result, the number of different policy sets that apply to these various modes and environments will grow in a combinatorial fashion. This will make it impractical to hard-code into the radio discrete policy sets to cover every case of interest. The accreditation of each discrete policy set would also be a major challenge. In the case of software-defined radios, it would require the maintenance of downloadable copies of software implementations of each policy set for every radio platform of interest.

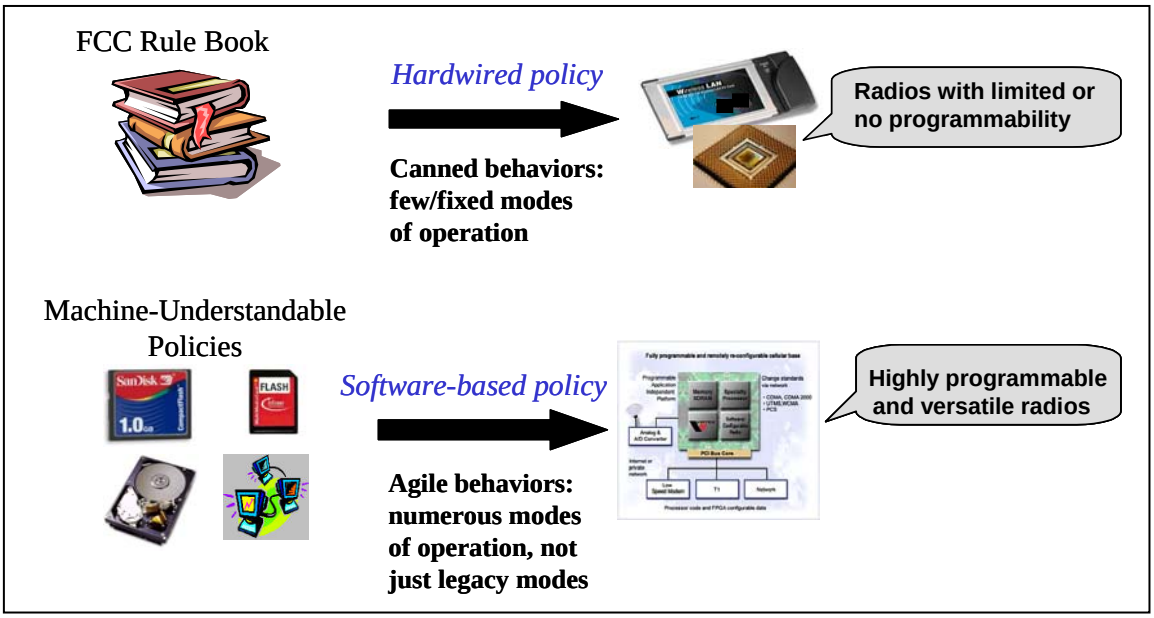


Figure 79: Machine understandable policies are necessary to exploit the emerging agility of devices and enable in-situ policy-based control of radio behaviors

We need a more scalable way to express and enforce policy. The complexity of accrediting policy conformance for XG radios and the desire for policy agility lead to the conclusion that XG radios must be able to read and interpret policy. We must, therefore, be able to express policy in a well-defined language framework.

Note: *This document addresses the means for expressing policies, to facilitate policy-conformant operation of XG radios. It is not intended to address other, equally important issues related to XG, for example, how XG radios will identify, select, coordinate, and utilize spectrum opportunities that are authorized by the policy. The XG Abstract Behaviors RFC (XGAB) will address some of these other issues.*

1.3 Requirements for an XG Policy Language Framework

Today spectrum policy is published (by a spectrum authority, such as FCC or NTIA in the U.S.A.) in a human readable form. This is entirely suitable for the current operational environment, in which humans interpret policies. As noted above, this will not be

adequate in future XG environments. In such dynamic environments, policy will need to be interpreted by the radio without human intervention. In particular we posit the following requirements for an XG policy language framework based on the XG vision [.

- ***Separation of policy and behaviors*:** It is a goal of the XG program to separate the policies that govern an XG system from the behaviors of the system. This will ease accreditation of XG systems by enabling regulators to accredit radios based on the ability to interpret policies correctly to obtain desired behavior, not (as today) by conformance of the implementation to specific pre-determined policies. These behaviors are discussed in the *XG Abstract Behaviors RFC*. In the absence of such separation, accreditation would need to be performed every time a policy was changed–even in a minor way.
- ***Adaptation to changing policies:*** XG radios must be adaptive to changes in policy, that result from evolution over time or changes in operational locale or use. Since XG technology is being developed in advance of policies governing dynamic spectrum management, policies are likely to be in flux for the near future. Consequently, users and spectrum regulators need a flexible method for reconfiguring policy within XG radios–one that does not require costly reengineering.
- ***Policy Consistency*:** As spectrum policy for adaptive spectrum access radios evolves and becomes more complex, opportunities for encountering inconsistencies and conflicts will increase. Verification mechanisms are needed to identify the interactions between policies in advance–before they are used–and to check the logical consistency of the policies. A policy language framework must facilitate such consistency checking and conflict resolution.
- ***New Capabilities*:** XG radios must be able to support a wide range of policies that correspond to their intended operational environments. These policies may depend on temporal, regional, or device parameters. Policies may permit XG underlay or overlay[12] of an existing spectrum use. Policies may also support authority delegation, enabling secondary spectrum markets where an authorized spectrum user may provide additional sub-policy inputs that allow use of the spectrum by non-primary users, including XG systems.

The ability of an XG system to interpret policy automatically is highly desirable–perhaps critical–for achieving the goals of the XG program. This ability in turn requires policy to be specified in a machine-understandable form. Hence, there is a need for a *language framework* to support machine-understandable policy. In the XG vision, policies (stored in a policy repository) will be supplied to XG devices in one or more "file(s)" on some media or downloaded from a network. The XG devices will read, understand, and conform to the policies reflected in the policy repository (or the appropriate portion thereof). Changing policy will require merely updating the policy repository and policy conformance will require that radios receive a current copy of relevant portions of the repository

Thus, deployment of XG-enabled systems will depend on the existence of a well-defined policy language framework; regulatory policies must be encoded using this framework; and XG devices must be able to interpret the policies so encoded. XG needs a *policy language framework* that defines language constructs for expression of policy, a machine understandable representation, and a concept of operations for their use.

Regulators will specify the policies that govern spectrum use by the radios. Regulators and policy makers may choose to encode and distribute machine-understandable

[12] See Appendix A for an explanation of spectrum underlay and overlay.

encodings of policies, or they may instead choose to certify encodings that are produced by other organizations. This latter scenario opens up the potential for competition and innovation in the encoding, processing, and system administration of machine understandable policies.

Regulatory policy will reflect the legal and/or contractual requirements for operating in a given jurisdiction. The designers or system administrators of an XG radio may provide additional policy inputs based on system-specific attributes of the radio to guide how it adapts to access the allowable spectrum. System policy governs radio actions within the boundaries specified by regulatory policies and can reflect administrative preferences or business strategy of the user's organization. In the military context, the commander's intent and the requirements of particular missions may govern the radio's actions by requiring radio-silence in some situations or offering additional guidance regarding the use of certain waveforms in other situations.

The policy language framework should allow both regulatory and system policy to be expressed, but must enable them to be enforced separately. The accredited portion of the radio must enforce the regulatory policy, while the system policy, which is enforced outside the accreditation boundary, will provide opportunities for innovation, including proprietary optimization techniques or other added value, by a radio manufacturer or service provider.

1.4 Evolution

This document provides an initial description of a policy language framework for expressing spectrum policy rules that addresses the above requirements. It is expected to evolve with changes in XG radio technology and policy. Furthermore, it is anticipated that a wide range of tools will be developed to facilitate the encoding of policy.

The base language, consisting of only the elements presented here, is not intended to be able to express all conceivable spectrum policies. It is a foundation from which policy administrators (who encode the policies that are specified by the policy makers) can start encoding policies. The language is designed using an extensible ontological framework, so this base language can be easily extended to be able to express additional spectrum policy rules and related concepts. For example, this document includes some frequently used policy parameters, but these are by no means an exhaustive list, and we expect that many more will be added to the language, as needed. We also expect the language to adopt and reuse generic ontologies as they are developed and standardized. For example, the language may adopt generic time or unit ontologies.

Note that the choice of a representation language does not mean that an originator of policy must directly use that language to express that policy. As an analogy, we note that few people now type HTML to create Web pages. Rather, they rely on fairly sophisticated graphical or scripting tools for Web page composition. These tools produce the HTML code that is eventually interpreted. In addition, other tools may be used to transform HTML to a form usable by particular vendors or platforms (e.g., Web clipping). Similarly, we envision that numerous tools will be developed to facilitate the writing and transformation of policy rules. In fact, some are available already. The shorthand notation introduced later in this document is a primitive example of such a

tool, and is intended primarily for the purposes of illustration and ease of understanding the language concepts described in this document.

1.5 Document Roadmap

The remainder of this document describes the XG Policy Language Framework in detail.

Section 2 provides an overview of the XG Policy Language Framework. It describes the concept of operations for both creating and interpreting policy in the XG Policy Language (XGPL). It provides a high-level view of the language itself and proposes the use of OWL a World Wide Web Consortium Recommendation for knowledge representation on the Semantic Web, to be used as the machine understandable representation of XGPL.

Section 3 provides a detailed ontology description of the language, including the concepts and keywords used in the language, and Section 4 defines the processing rules for the policy. Section 5 describes how the ontology presented in Section 3 can be extended. Readers who want only a broad overview of the language framework may skip these three sections.

Section 6 illustrates some key features of XGPL by encoding several (notional) policy excerpts. Section 7 concludes the document with a discussion of related policy work, and our plans to refine and extend XGPL in future revisions of this document.

Appendix A provides definitions of terms used in this document. Appendix B provides a list of hyperlinks (URLs) to an implementation of the XG policy language. Appendix C provides a notional policy along with an annotated encoding of the policy. Appendix D provides a formal specification of the shorthand notation used in this document. Finally, Appendix D also provides a mapping from the shorthand notation used in this document (and described in Sections 3.14 and 4.1) to the OWL representation of XGPL.

Note: *The examples shown in this document are intended only for illustrating language constructs. They are not intended to describe any past, present, or future policies of any spectrum authority.*

2 Overview

In this section we provide an overview of the XG policy language framework. At this point we note the distinction between the *policy language* and the *policy language framework*. We use the term *XG policy language* to refer to the language elements (including the syntactic and semantic structure of the language, and the domain-specific keywords), as well as the external (lexical) representation of the language. Overall, in this document we emphasize the language elements; we defer the lexical representation to the appendices. For this purpose, we make use of a shorthand notation to illustrate the concepts. We use the term *policy language framework* to include the policy language itself, the concept of operations of creation and use of machine understandable policies, and the computational logic, tools, and techniques for policy processing.

As an analogy for understanding our concept of policy language, consider that a medical treatise may, on the surface, appear to be just text written using English terms and syntax and using the Roman alphabet for representation. However, the medical terms and their interrelationships take on specific meanings within the domain, so that it is no longer useful to think of the treatise as merely some text written in the English language. Rather, the treatise is written in a language that is specific to the domain of discourse, but uses English as a basis of expression and representation.

Consider, for example, that the precise meaning of *acute* in medical terminology is different from its English meaning. Many other terms such as *arrhythmia* do not have meanings outside of the domain of medical discourse. Domain specific languages need to be extensible and enable the addition of new terms and interrelationships to their vocabularies. Furthermore, domain specific languages also typically include certain language constructs and idioms that are well understood by the users of that language.

It is important not to confuse the policy language with its lexical representation or its syntax. We will use OWL representation (based on RDF and XML) as a basis of expression—these are not to be confused with the XG policy language. Rather, the XG policy language consists of terms (along with their precise meanings), interrelationships between the terms, some constructs and idioms that are required to express policy, and the mechanisms by which the language can be extended.

In this section we describe a concept of operations–both how machine understandable policies are created and encoded by policy administrators, as well as how the encoded policy is used by policy-agile radios. We provide a brief overview of the features of the policy language and the policy processing logic. A detailed description of the language and the processing logic are provided later in Section 3 and 4 respectively. In Section 2.3 we examine the requirements for policy language representations and we propose an extensible, standards-based machine-understandable representation for policies.

A summary of features and benefits of the XG policy language framework is provided in Table 1. Examples illustrating some of these features are provided in Section 6.

Features	Benefits
Policy agility–not just hardware agility	Policy agility is the ability of a radio to interpret and conform to machine-understandable policy inputs; this feature is critical in order to exploit the emerging agility of devices and to allow *in-situ* policy-based control of radio behaviors. The current approach of hard-coding policy (in hardware or software) into agile radios would make accreditation extremely difficult, since it would require re-accreditation each time policy changes for any of a radio's operating modes. In addition, this feature enables the development of situational policies and time-dependent policies (e.g., an evolving "experimental" policy that is in effect for 3 months before being changed).
Well-defined accreditation boundary	The system components that check policy conformance are separate from those that are radio-specific and optimize performance. This feature enables regulatory concerns to be addressed separately from technology-specific optimizations, thus encouraging innovations in efficient spectrum sharing. As a result, the approach is more scalable; instead of needing to accredit each of n radios for each of m policies ($m \times n$), each radio needs to be accredited once, and each policy must be accredited once, and a single policy conformance checker must be accredited once ($m + n + 1$).
Use of international standards (W3C OWL)	A standards-based design assures maximum leverage of the work of others (e.g. development tools), and encourages broad international acceptance.
Extensible language	The ability to extend the policy language to handle new technologies and policies that are yet to be developed is critical for deployment. Without extensibility, the language would risk lagging behind technology, and hinder innovation. This feature serves to "future-proof" the language.
Multiple inheritance and polymorphism	The ability to inherit and extend policy language elements eases the process of encoding rules and eliminates the need to "re-code" similar rules or parameters separately.
Provable framework	By leveraging logical reasoning and theorem proving technologies (based on decades of research, and being developed for Semantic Web use) we can develop tools to perform *a priori* analysis of policy consistency.
Separation of policy from radio implementation via reusable abstractions	Policy does NOT tell the radio what to do–only what constitutes authorized use. Therefore policy can be specified in a vendor-neutral manner. Policy need not be specified separately for each device; instead policy is specified in terms of commonly specified abstract behaviors, the semantics of which are grounded separately for each device. This enables multiple developers and distributors (perhaps an open market) for machine-understandable policies and tools to process them.
Nesting of policies and sub-policy management	Nesting policy and sub-policy management provides the ability to inherit, extend, and combine policies from multiple authorities. It also enables incumbents (if authorized to reallocate spectrum) to manage policy for sub-leasing of spectrum. Multiple policies (e.g. based on authority, geography, time, device properties, or user base) can be specified.
Knowledge representation-based	Use of accepted knowledge representation (KR) techniques enables future language development to take advantage of advances in the KR field. Well-tested ontologies generated by others can be reused as well.
Separation of concerns via a rule-based approach for better maintainability	Use of a rule-based paradigm enables each rule to address an "aspect" of policy independently of others; rules can be incrementally added or deleted. This separation allows for more natural policy expression and is easier to maintain in comparison to rewriting programs of high complexity in a procedural language.

Table 5: XG Policy Language Framework–Features and Benefits

2.1 Concept of Operations

The concept of operations of the XG policy language framework describes how machine understandable policies are created and how they are used. First we will describe the actors involved, their roles in creating and using policy, and the elements of the policy that they use or control. Then we describe how radios can interpret and use the encoded policy.

2.1.1 Development of Machine Understandable Policies

Three different types of actors are involved in the definition and use of the policy language and the policy rules: language designers, policy administrators, and spectrum users. These actors will interact with the policy language and policy rules using tools appropriate for their role as illustrated in Figure 80 and described below.

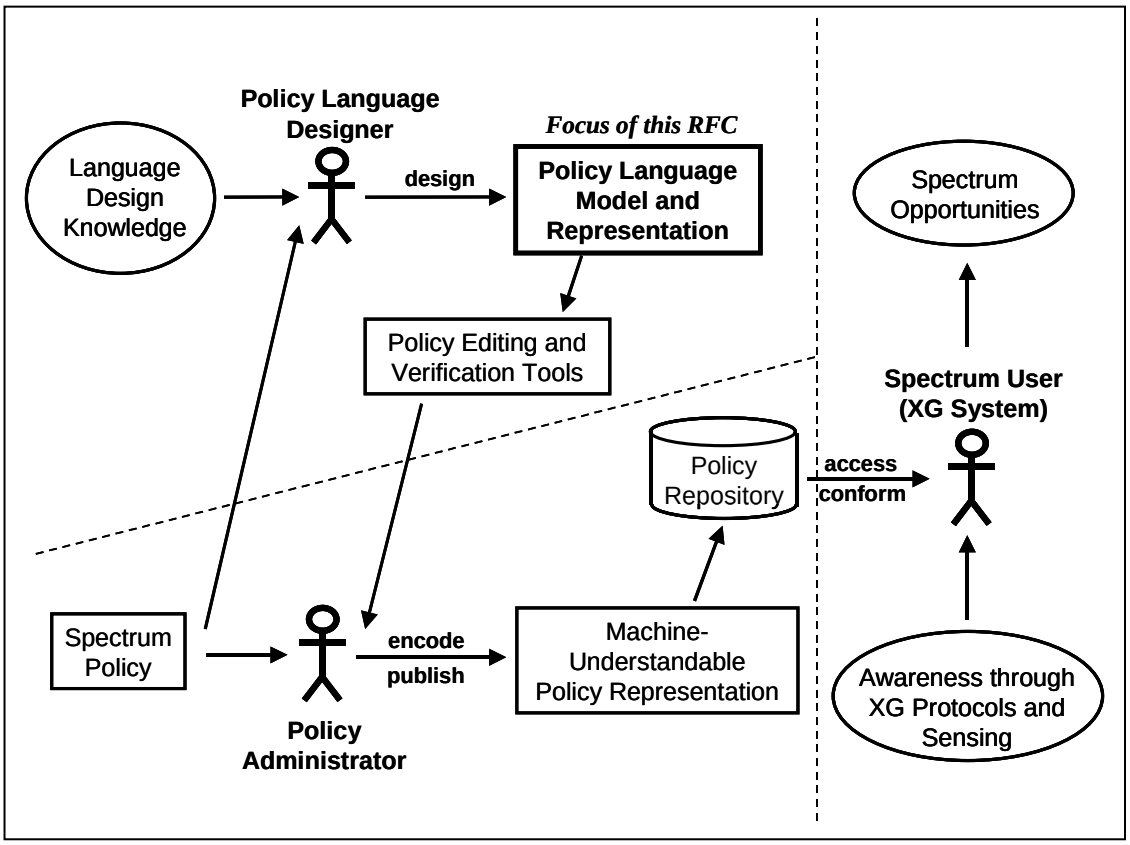


Figure 80: Policy Language Actors

Language designers create the *language model*, which defines the high-level objects of the language along with their semantics. This document describes an initial version of the language model. We expect that eventually a standards committee will draft future versions of the model. The language model is published at a well-known URI for policy administrators and policy users to access. The language designer may utilize tools such as UML modeling tools, tools to convert UML to a machine-understandable language (for example, see [BKK01]), and other visualization tools (yet to be developed) in order to facilitate the design process.

A *policy administrator* (who, as described in Section 1.3, is not necessarily the policy maker) is responsible for developing and encoding spectrum policy using the policy language produced by the language designers. The policy administrator makes the policy available to spectrum users. Administrators do not have to know all the details of the language presented here, as they will likely encode the policy by using a (perhaps graphical) tool, called an instance editor, which hides the notational complexity of the language. The administrator also uses analysis tools to verify in advance that the encoded policy has the desired effect and is consistent with existing policies.

In the short term, policy may be described first in English using engineering and legal terms that are commonly understood by the community of interest. Policy administrators will likely continue to use established procedures to interpret spectrum policy prior to encoding.

Given a policy expressed in English, the first step is to perform an analysis to determine how to structure the policy in a form best suited for machine understanding. Once this analysis is performed and the elements of the policy are mapped to the elements of the XG policy language, the policy can then be readily encoded in the XG policy language. In Appendix C, we present a notional policy example in English and show how to analyze the policy and encode it using the XG policy language by making use of a shorthand notation. Tools that enable the use of this shorthand notation (including a converter to XGPL) are available (see Appendix B). Other tools to validate the syntactic and logical correctness of the policy represented are planned. As this technology matures, we anticipate that a number of graphical and scripting tools will be developed to make the process easier.

A *spectrum user*, such as an XG system, must be able to use the policy to assess whether policy allows identified potential spectrum opportunities and to understand the constraints the policy administrators have placed on their use. The next section details how the spectrum user uses the policy.

2.1.2 Policy Usage

In this section, we describe a concept of operations for how a policy-agile spectrum user, such as an XG radio system, can use machine-understandable policy. Radios can use the machine-understandable policies to assure that its use of spectrum conforms to policy, as well as to modify radio behaviors in order to identify and utilize spectrum opportunities that are authorized. In this section, we will focus on the former, namely, how to assure policy conformance. The (planned) XG Abstract Behaviors RFC [XGAB] will describe XG radio behaviors related to acquiring and sharing spectrum awareness, as well as behaviors related to identifying, selecting, coordinating, and using available spectrum opportunities as authorized by policy.

Figure 81 is a logical functional block diagram of the policy usage concept of operations. The placement of these functional blocks in hardware or software is to be determined by the designers of each individual system.

There are four main components in this functional decomposition:

- ***Sensor:*** provides situational information to the radio about the spectral environment at a given location and time. This information is key to being situationally aware of spectrum opportunities enabled by policy. We note that the sensor outputs need not be limited to the RF spectral environment; the outputs could include a variety of other information (e.g. geo-location, temperature, and proximity to specific targets) that could be used as parameters for system policy.
- ***Radio Platform:*** provides the basic hardware and primitives of a host radio system that enables opportunistic use of spectrum (e.g., RF front end, DSP hardware, system software including the OS, middleware, and libraries, and primitives for networking protocols, waveform agility, and beam forming).
- ***Policy Conformance Reasoner:*** manages accredited policy information, which includes interpreting the policy language, and reasoning based on accredited policy (e.g., approved by a regulatory authority) and related background knowledge to determine whether the proposed spectrum use is policy-conformant or not; this key component to ensuring policy conformance is largely system independent and does not tell the radio platform or the system strategy reasoner what to do.
- ***System Strategy Reasoner:*** determines the system's strategy for opportunistic spectrum sharing under regulatory and system policy constraints; this reasoner is aware of system-specific optimizations and tradeoffs and has control over the radio platform. In Figure 81 we show the system strategy reasoner as logically separated from the rest of the radio. This highlights the potential for reusability.

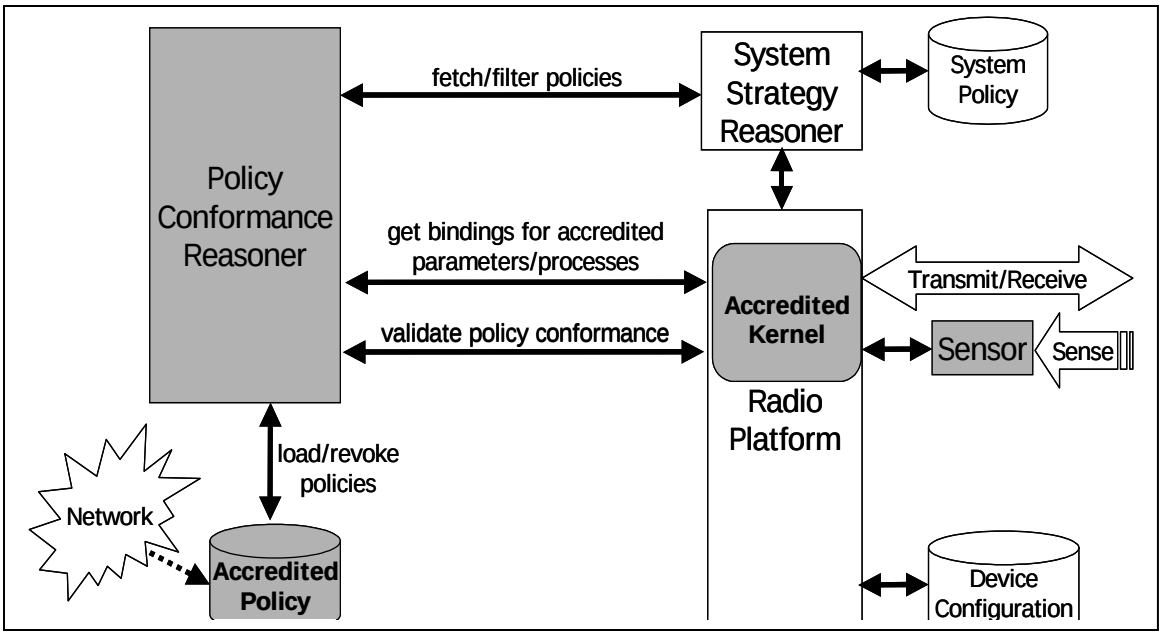


Figure 81: Policy-Agile Operation of XG Spectrum-Agile Radio

We envision that the radio platform includes an accredited kernel, within which it implements (grounds) accredited parameters and processes governed by policy. We further envision that only the policy conformance reasoner, the sensor, and the accredited kernel of the radio platform are within the accreditation boundary (shaded portions in Figure 81). Parts of the system that lie within the accreditation boundary are both necessary and sufficient to ensure policy conformance. As per the XG vision [XGV], system and protocol innovations including the system strategy reasoner are outside the accreditation boundary.

This architectural separation of the regulatory policy conformance function from the system dependent optimizations and tradeoffs allows the policy conformance reasoner to be reused and eases accreditation. Rather than separately accredit each of n radio configurations for each of m policy sets for a total of $m \times n$ operations, this approach reduces the required number of operations to m+n+1. The policy conformance reasoner must be accredited once, each policy set once, and each radio configuration once.

We describe the functions and interactions of the policy conformance reasoner in more detail here. The policy conformance reasoner performs three basic services:

- First, it loads (and possibly revokes) and interprets accredited policy instances, such as policies specified by a regulatory body, which are represented using the machine-understandable policy language.
- Second, it makes use of the policy structure to respond to queries (by a system strategy reasoner) to filter policies based on selection criteria (e.g. to a specified radio, or intended operating environment).
- Third, it determines whether or not the spectrum use proposed by the radio platform conforms to policy.

In order to explain the operation of the conformance reasoner, we provide a brief overview of the structure of policy rules in the XG policy language, described in more detail in Section 3.2. Each policy rule is composed of three parts: a selector description, an opportunity description, and a usage constraint description. A *description* is a Boolean predicate expression that involves regulated parameters and methods (for which bindings and groundings are provided by the radio platform) and additional policy parameters (for which bindings are provided within the policy itself). An *instance* is a set of bindings that can be used to determine the truth-value of a *description*.

A policy is *selected* and applied if the proposed selector instance satisfies the selector description. If the proposed opportunity instance matches the corresponding opportunity description, this provides *authorization* for the opportunity[13]. The proposed usage instance must fulfill certain *obligations*, i.e., it must satisfy the corresponding usage constraint descriptions. The use of selection, authorization, and obligation to structure policy rules is a paradigm that is used in other policy systems as well (e.g., [KAOS]).

[13]Positive and negative authorizations are supported.

We now present a notional sequence of operations using the functional decomposition presented in Figure 81. After loading any configuration information, the radio platform in conjunction with the sensor (potentially using a sensing strategy determined by system strategy reasoner) acquires awareness of its situation. The system strategy reasoner has access to the configuration, state, and awareness acquired by the radio platform through means that are specific to the system implementation. Based on system and regulatory policy and knowledge of the radio platform, the system strategy reasoner enables the radio platform to identify and characterize available opportunities and a suitable use of those opportunities. The characterization can result in, for example, binding values to relevant parameters.

Once an opportunity is identified and its intended use is characterized as a set of selector, opportunity, and usage constraint instances, the radio platform must present this set to the policy conformance reasoner for validation. In order to validate the instances, the policy conformance reasoner may need to obtain parameter bindings and invoke method groundings implemented within the accredited kernel of the radio platform. If the requested set of instances is validated, the XG radio may then use them to transmit.

The system strategy reasoner has the function of influencing radio behaviors in response to policy (and situational knowledge) in order to identify and utilize available spectrum as authorized. We note that there is significant scope for design diversity, innovation, and optimization within the system strategy reasoner, as well a potential for its reuse across several radio platforms. A simple system strategy reasoner may try only a limited range of opportunities and uses known to work with the radio and typical policies. A more sophisticated system strategy reasoner can include dynamic constraint solving capabilities; for example, it can query the policy conformance reasoner for applicable policy constraints (based on the state of the environment and the capabilities of the radio platform), and then determine a strategy to traverse the policy decision space efficiently to find good opportunities for the radio platform to use.

The design and specification of the interfaces between the radio platform and the policy conformance reasoner is a subject for future work.

2.2 Language Overview

This section provides a high-level overview of the XG policy language (XGPL) including the language elements described in more detail in Section 3, and the policy processing logic described in detail in Section 4. In this section we also present the requirements that motivate the use of OWL as the machine-understandable representation for the XG policy language.

2.2.1 Language Elements

An XG policy instance consists of a set of facts and rules. Facts define the policy and rules describe how to process the facts and hence how the facts relate to one another.

A policy rule is a statement of policy consisting of a set of facts, but not the rules for interpreting those facts. All other facts either further explain a policy rule or make

statements about a policy rule. Policy rules encode statements of policy, such as, “if peak received power is less than -80dBm then maximum EIRP is 10mW.” Most other facts either support the policy rules or refer to the policy rules. A policy rule links three facts: a selector description, an opportunity description, and a usage constraint description to describe a single statement in a policy, as illustrated in Figure 82.

The first fact in a policy rule is a selector description. This fact is used to filter policy rules to the set of rules that may apply to a given situation. The selector description points to one or more facts describing the authority that has jurisdiction over the policy, the frequency, time, and region the policy covers, and a description of the radio to which the policy rule applies. For example, a selector description may include filters such as “applies to operation in England” or “applies to operations in the broadcast bands”.

The second fact in a policy rule is an opportunity description. This fact provides an expression that is used to evaluate whether or not a given environment and device state represents an opportunity for this policy rule. For example, it can express opportunities such as: “peak received power is less than -80dBm” or “if a beacon is heard at 823MHz.” The opportunity description is not evaluated unless the selector description from its policy rule matches. If a potential opportunity, called an opportunity instance, matches the opportunity description, a flag in the policy rule indicates if this is a valid or invalid opportunity. A valid opportunity indicates transmission that conforms to the usage constraint description is permitted. An invalid opportunity indicates that transmission is not permitted.

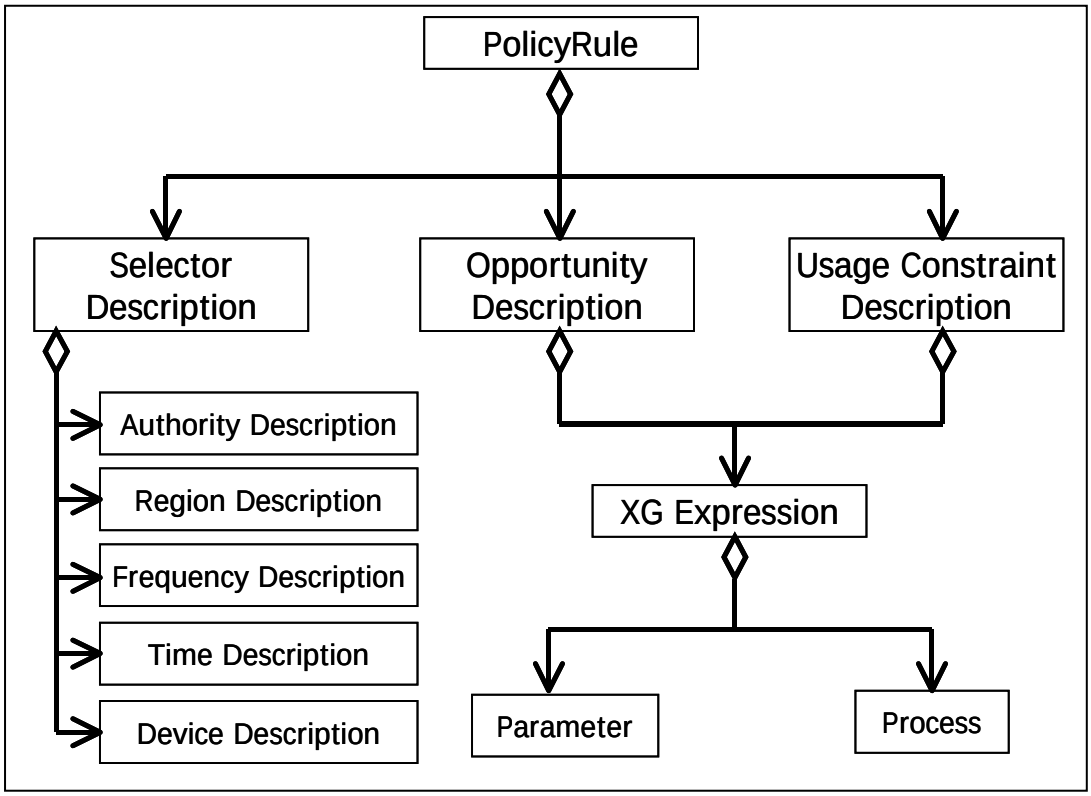


Figure 82: Structure of Policy Facts

The final fact in a policy rule is a usage constraint description. This fact provides an expression that constrains the radio behavior, such as the emissions permitted and the corresponding sensing requirements when using the opportunity described in the policy rule. For example, it can express a usage such as: “transmit with a maximum EIRP of 10mW” or “maximum continuous on time must be 1 second and the minimum off time must be 100msec.”

2.2.1.1 Extensible Semantics of Policy Parameters and Processes

As new radio platforms and policy sets are developed, they will have capabilities that were not envisioned when the policy language was developed. However, we will need to represent facts about these radios to fully describe selector, opportunity, and usage constraint descriptions that constitute policy. The policy language supports two generic constructs, namely, parameters and processes to represent policy concepts.

Parameter facts define all the values that are in the policy. They include values such as frequency bands, power levels, times and geographic areas. Parameters may be bound if the value is known, or unbound if the value is not yet known, such as a value that the radio platform is expected to bind.

Some policy rules will require the radio to perform certain functions to provide the information necessary to match an opportunity or usage constraint description. These are described using *process facts*. The process fact describes inputs and output parameters for the function (analogous to a function prototype in a programming language such as C) and possibly expressions constraining the relationship between the inputs and outputs. The radio is responsible for providing an implementation of the process. If a radio doesn't support a needed process, then it cannot use the opportunity that the policy rule describes.

In addition to these generic constructs, we have organized some of the key concepts in this domain into *ontologies* to provide a usable foundation. This organization is described in more detail in Section 3. These ontologies include a structure of authority and delegation, frequency classifications, geopolitical regions and spatial descriptions, time, device capability descriptions, environment and device state descriptions, physical quantities, and units.

For example, an authority is an entity that has jurisdiction over some frequency, region, time, and set of devices and is authorized to create policy for that jurisdiction. An authority may be, for example, a regulatory agency or a primary user who is authorized to lease their spectrum to other users. A fact defines the authority and states its jurisdiction.

Some policy rules may apply only to a radio with a specific set of capabilities, either because the policy rule is designed for a specific type of radio or because it requires a particular set of parameter types or processes for the policy rule to be evaluated (e.g., supports geo-location, implements database access function, or maintains a history of parameter values). Such information is captured in a device description fact.

2.2.1.2 Meta-Policies

In absence of any other information, the set union of the usage constraint descriptions from all the policy rules that represent valid opportunities apply. However, meta-policy facts may state relationships between policy rules that modify this logic. We have included three types of meta-policy facts: grouping, precedence, and disjunction.

Grouping simply creates a named set of policies so they can be referenced as a group. The group may be described either by explicitly listing the member policy rules or by creating an expression that describes the policy rules that are members of the group.

Precedence provides information on how to interpret conflicting policy rules. Two policy rules that have matching selector descriptions may also have opportunity descriptions that match a particular opportunity instance. If both these opportunities represent a valid opportunity the usage constraints for both policy rules apply. However, if the policy rules disagree on whether the opportunity instance is valid, there is a conflict and the rule with the higher precedence is applied. Precedence may be defined between two policy rules or two policy groups. If one policy group has higher precedence than another, then all its member policy rules have a higher precedence than the member rules of the other group. Finally, two policy groups may be disjunctive. If policy groups are disjunctive, then the policy rules in one or more of those otherwise applicable groups can be selected for application.

2.2.2 Policy Processing Logic

Policy processing, within the policy conformance reasoner for example, requires a set of rules describing how to process and interpret the policy facts. These rules govern the selection of policy rules that match a given selector instance; the selection of a policy rule that represents a valid opportunity given a selector instance and an opportunity instance; and the conformance of a given selector, opportunity, and usage constraint instance to the policy set. These rules also govern meta-policy processing as well as the decorrelation of selector instances described in Section 4.

The processing rules allow regulators to create policy rules that build on existing rules. Unless modified by a meta-policy, the union of all usage constraints from policy rules that represent valid opportunities is required for transmission to occur in that opportunity. Therefore, policy administrators do not have to enumerate policies for each band but may, instead, create broad policies and refine them where necessary.

For example, if a policy administrator wanted to constrain emission power in the television broadcast bands to –10dBm where an opportunity was detected, but additionally restrict emissions to have a 1MHz bandwidth in the frequencies[14] represented by channels 58-63, only two policy rules need to be written. The first rule would set the –10dBm constraint on all television channels and the second rule would set

[14] Supporting information such as the mapping of channel numbers to frequencies must exist elsewhere in the policy.

the bandwidth constraint on channels 58-63. If a radio wanted to transmit on channel 61, it would have to conform to both emission constraints, as both policy rules apply. At a later stage, regulators may easily add a third policy rule that constrains new radios of a particular type from transmitting on channel 59 in selected regions, without modifying the other two rules.

We believe that the union of simpler rules (logical implications) scales well and offers flexible management of policies. In contrast, the use of procedural structures such as if-then-else conditions and while-loops can result in brittle policy structures over time as policy evolves. For example, a small change in policy can result in a large number of changes that must be reflected across several if-then-else branches or require the entire policy to be rewritten. Furthermore, an if-then-else structure can artificially constrain the order in which conditions are tested by a particular radio.

2.3 Language Features and Representation

A standard representation for the XG policy language is necessary so that regulators can encode policies in one language and all XG radios understand the encoded policy. This section discusses the features required for representing XGPL, provides an overview for some representations considered and introduces the OWL Web Ontology Language, the standard representation used for XGPL.

2.3.1 Requirements

Spectrum policies have a complex structure with many dimensions and layers of exceptions that are difficult even for human interpretation. In selecting a language, one must ensure that the language is capable of capturing and potentially simplifying a number of aspects of this complex structure, including:

- *Inheritance*. Spectrum policy is very large and complex. The property of inheritance helps manage this complexity by enabling policy rules and properties to extend others and reduce the need for enumeration. For instance, rules for the 2.4GHz unlicensed band can inherit and extend rules for general unlicensed use.
- *Reification (rules about rules)*. Policy rules may make statements about other policy rules. For example, we can make a policy rule governing when or where a set of policies will apply.
- *Inference (derivable rules)*. There are rules that may not be explicitly stated, but follow from two or more rules. For instance, given that TV channels are 6 MHz wide, and that a policy exists to permit XG devices to transmit within a locally unallocated TV channel, one may infer that the maximum bandwidth for an XG device using an unallocated TV channel is 6 MHz. More common languages such as XML and IDL, while perhaps more straightforward for humans, do not facilitate such inference.
- *Extensibility*. It is imperative that the policy language be extensible in its vocabulary, structure, and semantics so that the language can adapt to express

new types of policies as spectrum policy requirements change. Additionally, different countries may be concerned with different types of policies or different signal parameters, and it is crucial that the language be extensible to include their requirements.

- *Maintainability*. Policy will evolve over time. Small changes in policy should require only small, localized changes in the policy encoding and should not have ripple effects throughout the policy. Additionally, the language should make it easy to select policies that apply to a particular situation (i.e. time, location, frequency bands, device type). This argues for a declarative approach based on facts and rules instead of a procedurally based language as exceptions. We discuss this more below.
- *Scalability*. The policy language must scale to a wide range of devices that may have very different levels of resources available. The policy specification must be amenable to processing on devices with small memory and computational resources.
- *Standards*. Using a standard language representation is preferred as it enables reuse of libraries, applications, and tools previously developed. These tools reduce the cost of developing and using the language and increase interoperability between different tools. It is also expected that using a standard representation will facilitate the international acceptance of XGPL.

The language should be a declarative language based on facts and rules instead of a procedural language. One reason is maintainability as mentioned above. In a procedural language, policy would become a nested set of exceptions–many layers of `if-then-else` clauses. This creates two problems. First a small change in the policy may then affect many exception clauses, making it difficult to update policy. Second, the order in which exceptions are defined necessarily optimizes the policy for particular parameters. If a radio is concerned with other parameters, it will have a less optimal search. Additionally, regulatory policy does not tell the radio what to do; it only defines what constitutes authorized use of the spectrum. However, enforcing a policy may require the results of a function that may be implemented in the XG radio. Using a declarative language, the policy can describe the function and specify rules based on its inputs and outputs without specifying the particular implementation of the function.

2.3.2 OWL

The World Wide Web Consortium's Web Ontology Language [OWL] is a machine-understandable, semantic markup language. It is an extension of the Extensible Markup Language (XML) and the Resource Description Framework (RDF). OWL is based on DAML+OIL, a combination of efforts in the US (DAML–the DARPA Agent Markup Language) and the European Union (OIL - the Ontology Inference Layer) to create a machine-understandable semantic markup language. OWL provides a rich language for

representing an ontology[15], that is knowledge about the interrelationship of objects, in a manner that allows a machine to make inferences.

OWL builds upon several XML layers:

- *XML Document Type Declaration (DTD) and Namespaces:* XML is a popular portable encoding for data exchange as it offers separation of style, syntax, and content. The Document Type Declaration (DTD) represents the basic XML layer. It can encode the grammar for a document. Since the grammar is defined separately from the content, general-purpose applications may be built that do not embed a particular grammar. *XML Namespaces* allow the grammars to have constructs with universal names, whose scope extends beyond their containing document.
- *XML Schemas.* The next layer, XML schemas [XSD] provide support for datatypes and more structure than DTDs. Additionally, an object-oriented schema, SOX is available as well as alternative XML schemas such as RELAXNG [RLXNG]. However, XML schemas are still limited as they support only data representation; semantic knowledge must still be embedded in an application.
- The *Resource Description Framework (RDF)* in conjunction with *RDF Schema (RDFS)* can capture semantic knowledge and is extensible. However, it does not support inference. As mentioned earlier, we need a language for inferring information about the policy from the rules provided.

OWL builds upon the XML layers[16] described above and incorporates insights from knowledge representation research. OWL enables the use of powerful processing tools (such as inference engines and theorem provers) that are being developed for the Semantic Web.

There are three variations of OWL. OWL DL provides the maximum expressiveness of OWL while providing well-understood complexity characteristics of description logics. OWL Lite is a subset of OWL DL that trades off expressiveness for processing complexity. For example, it restricts some of the constraints available in OWL DL, such as limiting cardinality restrictions to be 0 or 1, or the restriction of property constraints locally to a class. OWL Full supports the same language constructs as OWL DL, but relaxes some of OWL DL's restrictions that provide its computational guarantees. XGPL will use OWL DL as the computational guarantees are required and it needs more expressiveness than OWL Lite provides.

[15] An “ontology” is the knowledge about the relationships between objects. However, OWL also uses the term “ontology” to refer to an OWL file that encodes ontological information.
[16] For a comparison of the capabilities of XML, RDF(s), DAML+OIL and OWL, see [COMP]

2.3.3 Why OWL for XG?

OWL provides a crucial set of features that match the requirements for representing the complex structure of spectrum policy. Specifically, OWL is extensible, and it supports reification, inference, and several object-oriented features such as multiple inheritance.

Powerful tools are being developed to process OWL that build upon earlier research in the knowledge representation, logic, and theorem proving communities in order to support the semantic web.

We chose to focus on markup languages for several reasons. First, the automatic conversion from a highly structured markup format to other non-markup formats is relatively easy; the reverse transformation is typically more difficult and non-uniform. Markup languages and tools to process them are widely adopted around the world and are continually evolving, as is evidenced by the Web. These Web markup standards satisfy the need for a representation that is suitable for cross-platform information exchange and processing across nations and organizations.

In an earlier work, Mitola and Maguire [MITOL] investigated a wide variety of languages to represent policy for cognitive radios. They concluded, as we do, that a knowledge representation approach is the most suitable for radio policy. They recommended the development of a new language based on the Knowledge Query and Manipulation Language (KQML), a language that has limited exposure outside the knowledge-based systems community. Since the time of their study, the World Wide Web Consortium has developed the OWL Web Ontology Language. Using a language that is a W3 Consortium Recommendation offers potentially wider adoption.

OWL's predecessor, DAML, has also been demonstrated to be an effective technology in other complex policy domains. The KAoS Domain and Policy Services [KAOS] project has successfully used DAML to represent distributed logistics policies for the DARPA UltraLog program [ULTLG].

In summary, OWL provides the features required to implement a machine-understandable semantic model. This enables the building of generic applications that depend on knowledge of the semantics of the content, including generic theorem provers and reasoning engines that enable deductive inference. OWL has the ability to represent the syntax of spectrum policy and the complex knowledge embedded in it. Additionally, OWL provides the ability to deduce policy constraints that are not explicitly stated for each specific case, but rather inferred from more general statements. Finally, we note that OWL is a Recommendation of the World Wide Web Consortium. Thus, we believe that OWL is an excellent long-term solution for the XG policy language.

Therefore, we provide an OWL representation of the XG policy language ontologies. We further show how to use these ontologies to specify machine-understandable spectrum policy separately from the implementation of the policy within particular radios.

3 Policy Ontology

This section describes the XG Policy Language ontology in detail. It describes all the classes in the language and their relationships. The URLs of the OWL ontology files that define the language are listed in Appendix B.

We expect that, in the future, some of the ontologies presented here will be partially or completely replaced by generic ontologies as they are developed and standardized. We reference some ontologies that are currently being developed and may be used in the future.

3.1 Facts, Expressions, and Rules

The XG policy language framework is built around three basic constructs: facts, expressions, and rules. Policies encoded in XGPL consist of a set of facts and expressions. Rule constructs are used to specify processing logic for policies.

3.1.1 Facts

Facts are OWL statements that describe the policy concepts. The domain, the XG radio, grounds the semantics of the facts. For convenience, two semantic extensions to OWL—expressions and rules—are provided in the XG policy language. These extensions are also represented using OWL.

There are several types of OWL statements: classes, properties, restrictions, Boolean class expressions, and individuals. Classes provide an abstract definition of a group of resources with similar characteristics. Classes can be defined in terms of other classes (for example, as a subclass of or as an equivalent class to another class) and restrictions on properties. Properties may either be object properties that relate an individual (instance) to another individual, or data-type properties that relate an individual to a data-type value. Restrictions limit properties, including the values that properties may have and the cardinality of a property. Boolean class expressions allow a class to be described as an intersection, a union, or a complement of other classes. Individuals are specific instances of a class; instances are identified by a URI and may have particular values associated with each property of the class. For a more detailed description, informative tutorials, and the current version of the OWL Recommendation, please refer to the W3C OWL website [OWL].

The specification of an XG policy involves defining a set of individuals that express the policy and conform to the ontologies presented in this document. The XG policy language ontologies themselves are also described using OWL statements.

3.1.2 Expressions

Expressions in the XG policy language are used to define an opportunity, a usage description, or to define membership in a policy group. Each XG expression is a predicate expression that returns a Boolean value. There are four kinds of predicates.

- Logical operators: The logical operators comprise the top-level predicates. These operators include: `and`, `or`, `not`, and `exists`.

- Relational operators: Relational operators, including interval operators, comprise the second-level predicates. These operators include the scalar comparison operators: <, >, >=, <=, =, `eq`, and the interval comparison operators: `before`, `after`, `within`, `contains`, `overlaps-start`, `overlaps-end`, `just-before` (before and overlaps-start), `just-after` (after and overlaps-end), `at-start-of` (within and overlaps-start), `at-end-of` (within and overlaps-end), `starts-with` (contains and overlaps-start), `ends-with` (contains and overlaps-end), `overlaps` (within and contains and overlaps-start and overlaps-end).
- Other predicates. Any additional domain-specific functions required must be declared as a `Process` as discussed below and invoked using the `invoke` predicate. The `invoke` predicate returns a Boolean value to indicate whether the invocation succeeded or not. Processes can be defined for arbitrary functions such as mathematical operators, signal processing primitives, and other system operations such as the sensing of signals, and database retrieval. Such processes must be grounded by the radio platform to be used.

If automatic constraint solving is desired based on the encoded policy, then care must be exercised in the use of XG expressions, especially the use of universal and existential quantifiers, as well as the use of domain predicates that are computationally expensive.

3.1.3 Rules

Rules are statements that describe the logic for interpreting and processing policy. They have the form: *condition-implies-action*. If a fact matches the set of expressions that form the condition, then the function expressions in the action are asserted. The XG policy language includes a rule ontology. A description of the ontology is included in the Backus-Naur Form (BNF) of the shorthand notation in Appendix D. In particular, the `defrule` construct included in the BNF maps to the `Rule` class provided in the XG rule ontology.

Rules are not needed for encoding a policy; however, they are needed to express the logic for processing and interpretation of the policy rules. As such, the rules are background knowledge, included as part of the ontologies. The rules for processing and interpreting the XG policy language are described in Section 4.

Note that in the future, the XG policy language may instead use the OWL Rules [SWRL] when this ontology and the tools to process it are further developed.

3.1.4 Shorthand Notation

For ease of representation we will introduce a shorthand notation for expressing facts and expressions. A shorthand notation for expressing rules is described in Section 4.1. We use the shorthand notation since it allows us to illustrate the concepts more readily in the examples. The shorthand has a one-to-one correspondence to OWL. Appendix E provides a mapping of the structures and keywords presented in this section to their OWL equivalents.

The shorthand notation is based on an existing rule-based knowledge representation environment [CLIPS].

A Class is defined as follows:

```
(deftemplate <class-name>
  (slot <slot-name>)
  (multislot <slot-name>))
```

For example:

```
(deftemplate FrequencyDesc (slot id) (multislot frequencyRanges))
```

The `<class-name>` represents the name of the class and the `<slot-name>` represents the name of a property of the class. A `slot` may only have a single-valued property as `<slot-name>` while a `multislot` may have multiple-valued properties as `<slot-name>`'s. All XGPL `deftemplate`s have one slot, `id`, which must be explicitly included. It is used to identify an instance of the class.

An instance of `FrequencyDesc` can be represented as follows:

```
(FrequencyDesc (id MyFrequencies)
 (frequencyRanges Range1 Range2 Range3))
```

This indicates that `MyFrequencies` is an individual that belongs to the class `FrequencyDesc`, with the property `frequencyRanges` assigned the values `Range1`, `Range2`, and `Range3`, which are described by other statements, for example:

```
(FrequencyRange (id Range1) (min 3.4) (max 3.5) (unit GHz))
```

In addition to classes, the shorthand notation supports Boolean expressions in prefix notation:

```
(<predicate> <arguments>+)
```

To illustrate, we can express the truth value of “9 is greater than 8” as follows:

```
(> 9 8)
```

The Boolean expressions are part of an XG expression that is enclosed in double quotes. To use a process in an XG expression, the `invoke` predicate is used. The first argument it takes is the name of the function being invoked. This can be followed by a sequence of name value pairs for input and output parameters.

For example, we could have an XG expression that states that the difference between the maximum transmit power and the maximum received power must be greater than a specified power `Power20dBm`. Say we have declared three device parameters to be bound by the XG radio, `PowerParam1`, `PowerParam2` and `PowerParam3` that represent a maximum transmit power, maximum receive power and the output of the process, respectively. We may also define a process called `Sub`, the mathematical function of subtraction, with input parameter names `MaxTransmitPower` and `MaxReceivePower` and a single output parameter named `PowerDiff1`. Now we can state the following XG expression:

```
(and (invoke Sub MaxTransmitPower PowerParam1
                 MaxReceivePower PowerParam2
                 PowerRatio PowerRatio1)
     (> PowerRatio1 Ratio20dB))
```

All the policy facts in the shorthand notation are wrapped in a `deffacts` structure that is defined as follows:

```
(deffacts <deffacts-name>
     (<deftemplate-name>
        (<slot-name> <constant>+)*)*)
```

So we could have a set of facts defined as follows:

```
(deffacts my-policy-facts
  (FrequencyDesc (id MyFrequencies)
     (frequencyRanges Range1 Range2 Range3))
  (PolicyRule (id P1) (selDesc S1) (deny FALSE)(oppDesc O1)
              (useDesc U1))
)
```

The remainder of the shorthand notation is the set of keywords for the classes, slots, and predicates. These will be developed as part of the ontology in the remainder of this section. The notation also supports rules. A BNF description of the shorthand notation may be found in Appendix D.

3.2 Structure of XG Policy

An XG policy rule is a fact that specifies one aspect of a policy. The set of XG policy specification facts is given by PSF = $\{P_i: <S_i, O_i, U_{i,} deny_i>\}$ where:

- S_i is a selector description
- O_i is an opportunity description
- U_i is a usage constraint description
- $deny_i$ indicates whether or not matching the opportunity descriptions means that it is an invalid opportunity.

Accordingly, the structure of the `PolicyRule` class is as follows:

```
(deftemplate PolicyRule
      (slot id)
      (slot deny)
      (slot selDesc)
      (slot oppDesc)
      (slot useDesc)
)
```

The respective ontologies for selector description, opportunity description, and usage constraint description are described later. Rules (described later) interpret and process the policy facts and potentially generate additional policy facts.
The `deny` property allows policies to explicitly rule out opportunities at the logical level, thereby avoiding the need to process usage constraint descriptions corresponding to denied opportunities. We explain how this property is used later in Section 4.3.

3.2.1 Policy Selector

Selectors classify policy rules for easy filtering of potentially relevant policy. A policy selector fact is a five-tuple defined as $S_i:<A_i, F_i, R_i, T_i, D_i>$ where:

- A_i is an authority description
- F_i is a frequency description

- R_i is a spatial region description
- T_i is a temporal description
- D_i is a device description

Additional descriptions may be useful for policy filtering and will be considered in future revisions of this document. Two such descriptions are a user description that identifies the class of users for which the policy is intended (e.g. US Federal Government users) and a service description that identifies the service (e.g., land mobile, broadcast) to which the policy refers.

The `SelDesc` class is therefore defined as follows:

```
(deftemplate SelDesc
      (slot id)
      (slot authDesc)
      (slot freqDesc)
      (slot regnDesc)
      (slot timeDesc)
      (slot devcDesc)
)
```

The corresponding classes for each slot are:

- `Authority` – An authority and its jurisdiction is defined as:

```
(deftemplate Authority
      (slot id)
      (slot polAdmin)
      (slot freqDesc)
      (slot regnDesc)
      (slot timeDesc)
      (slot devcDesc)
)
```

  Where the corresponding classes for each slot are `PolicyAdministrator`, `FrequencyDesc`, `RegionDesc`, `TimeDesc`, and `DeviceDesc`, respectively. The authority ontology is described in detail in Section 3.5.

- `FrequencyDesc`– A frequency description is a list of `FrequencySpecification`s. It is defined as follows:

```
(deftemplate FrequencyDesc
      (slot id)
 (multislot frequencyRanges))
```

  The frequency ontology is described in detail in Section 3.6.

- `RegionDesc` – A region description is a one or more `GeographicSpecification`s. It is defined as:

```
(deftemplate RegionDesc
      (slot id)
 (multislot region))
```

  The region ontology is described in detail in Section 3.7.

.`TimeDesc` – A time description is a list of `TimeIntervals`. It is defined as:

```
(deftemplate TimeDesc
      (slot id)
 (multislot time))
```

The time ontology is described in detail in Section 3.8.

- `DeviceDesc` – A device description includes a device type description `DeviceTyp`, and a device capabilities description `DeviceCap`. It is defined as:

```
(deftemplate DeviceDesc
      (slot id)
      (slot deviceTyp)
      (slot deviceCap))
```

The device ontology is described in detail in Section 3.9.

The ontologies in the following sections further develop the values that each of these descriptions may specify.

A selector description can be viewed as a region in a multidimensional space (of frequency, space, time, and device parameters) tagged with an authority that has jurisdiction over that space. A selector instance also represents a region in that multidimensional space. Given a selector instance S_x, and a selector description S_i, exactly one of the following facts can be asserted:

- `(disjoint` S_x S_i`)`–The region of the selector instance is completely outside the region defined by the selector description.
- `(equivalent` S_x S_i`)`–The region of the selector instance is the same as or is completely contained within the region defined by the selector description.
- `(correlated` S_x S_i`)` - The region of the selector instance intersects the region defined by the selector description. An equivalent selector is also correlated.

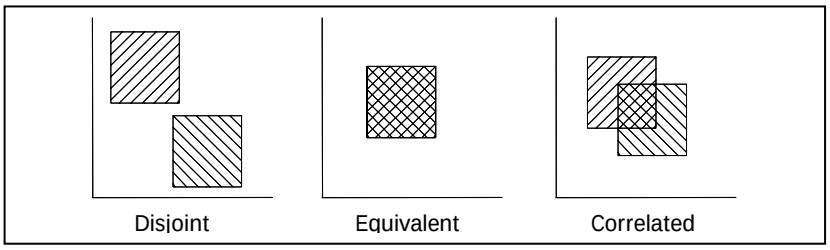


Figure 83: Two-dimensional Illustration of Disjoint, Equivalent, and Correlated

This is illustrated in Figure 83. If a selector instance is correlated, then it must be partitioned into other instances such that each instance is correlated with a single selector description. This process is called *decorrelation.*

3.2.2 Opportunity Description

An opportunity description is a condition expression based on parameters describing device and environment state that is used to determine if a valid opportunity exists. Such conditions could include, for example, the presence or absence of particular incumbent

signals, the use of a feature-matched receiver, and a threshold for sensor sensitivity. An opportunity description includes the property `xgx` that specifies an XG expression based on parameters to which the radio must bind values. The opportunity description is defined as:

```
(deftemplate OppDesc
      (slot id)
      (slot xgx)
)
```

Given an opportunity instance O_x, and an opportunity description O_i, we can assert exactly one of the following facts.

- `(matches` O_x O_i`)` - The parameters in the opportunity instance satisfy the `xgx` expression in the opportunity description.
- `(notGermane` O_x O_i`)` - The opportunity doesn't match so is not relevant to this description.

Whether a matching opportunity instance must be allowed or denied is indicated in the `deny` slot of the policy rule that uses the instance description. If `deny` is false, a matching opportunity instance must be allowed; if it's true, then it must be denied. Positive and negative authorizations, used commonly in policy literature, correspond to this allow and deny.

3.2.3 Usage Constraints Description

A usage constraint description is a constraint expression based on variables describing device and environment state. Using our language, a usage constraint description can express a variety of constraints on spectrum use. For example, constraints can be placed on the maximum transmit power within the emission frequency band, the duty cycle, the energy in the harmonics (splatter due to amplifier distortion), co-channel interference, and the field-of-view and side lobe energy when using directional antennas.
As with an opportunity description, the usage constraint description contains a property `xgx` that specifies an XG expression based on parameters to which the radio must bind values. A usage constraint description is defined below:

```
(deftemplate UseDesc
      (slot id)
      (slot xgx)
)
```

Given a set of usage constraint descriptions $\{U_i\}$ and a proposed usage instance U_x, we can assert exactly one of the following facts.

- `(solutionOf` U_x $\{U_i\}$`)`
- `(notSolutionOf` U_x $\{U_i\}$`)`

3.2.4 Meta-Policy

Meta-policies are facts that relate two or more policy rules to each other and change how they are processed. Three types of meta-policies are included in the XG policy language: grouping, precedence, and disjunction. This section discusses the specification.

3.2.4.1 Grouping

Grouping is a means of grouping policy rules into a single set, and assigning a single name to the set of policy rules. Policy groups may be defined in one of two ways. The first is to enumerate all the rules that are part of the group. In the second, an XG expression can be defined and all policy rules that match that expression are part of the group.

The class `PolicyGrp` specifies a policy group. It contains three properties: `equalPrecedence`, `polMembers` and `xgx`. Only one of `polMembers` or `xgx` may be present in an instance of `PolicyGrp`. The property `polMembers` contains a list of policy rules that are members of the group. `xgx` contains an XG expression that defines membership in the group. The property `equalPrecedence` indicates whether or not all the policy rules in the group have the same precedence.

```
(deftemplate PolicyGrp
      (slot id)
      (slot equalPrecedence)
 (multislot polMembers)
      (slot xgx)
)
```

Given a policy, P_i, and groups, G_k and G_i, we can assert the following facts:

- `(isMemberOf` P_i G_k`)`–if P_i is listed in `polMembers` or matches the expression in `xgx`
- `(isSubsetOf` G_i G_k `)`–if all the policy rules in G_i are also in G_k.

3.2.4.2 Precedence

Precedence is the mechanism used to determine which policy rule applies if multiple policy rules conflict on whether an opportunity instance is valid or not. In XGPL the precedence of policy rules can be stated explicitly.

The `PolPrecedes` fact specifies the precedence between two facts. The properties `left` and `right` indicate the two policies, where `left` precedes `right`.

```
(deftemplate PolPrecedes
      (slot id)
      (slot left)
      (slot right)
)
```

We can assert that policy rule `(precedes` P_x P_j`)` from this `PolPrecedes` fact

`(PolPrecedes (left` P_x`) (right` P_j`))`

There are two properties of precedence that should be noted. First, if `(and (precedes` P_x P_j`) (precedes` P_j P_x`))` are asserted then P_j and P_x have the same precedence. Second, precedence is transitive, so if `(and (precedes` P_i P_j`) (precedes` P_i P_k`))` are asserted that implies `(precedes` P_i P_k`)`.

Similarly, we can explicitly state precedences between groups with the `PolGrpPrecedes` fact. The properties `left` and `right` indicate the two groups, where `left` precedes `right`.

```
(deftemplate PolGrpPrecedes
      (slot id)
      (slot left)
      (slot right)
```

```
)
```

For example:

```
(PolGrpPrecedes (left G_i) (right G_k))
```

3.2.4.3 Disjunction

Two or more groups of policies may have policy rules with equivalent selector descriptions and opportunity descriptions that may both be matched by a particular opportunity instance. Barring a meta-policy, the usage constraints from all policies (that allow the opportunity) from all groups will apply conjunctively. However, the policy writer may want to group constraints disjunctively so that the XG radio needs to apply all constraints from only the subset of groups under which it chooses to operate.

The `DisjunctGrps` fact specifies that all policies from a subset of the groups enumerated in `polGroups` can be enforced at one time. The fact is specified as follows:

```
(deftemplate DisjunctGrps
      (slot id)
 (multislot polGroups)
)
```

3.3 Parameters

Many of the sections above describe ontologies that involve physical quantities. This section describes the ontology for expressing those quantities in a uniform manner.

Each physical quantity is a subclass of one of the parameter classes discussed in this section. Each parameter used in a selector, opportunity, or usage constraint description must be declared as a fact in the policy. The declaration must include the name of the parameter. Some parameters will be bound to a specific value as part of the policy. For example, the policy may declare a parameter that refers to -100 dBm. Other parameters are declared without values and the values must be bound by the XG radio platform. Finally, some parameters, such as parameters in process definitions, are defined without values, but the values are bound by the policy processing.

Each parameter class below is a subclass of the `Parameter` class. The parameter class has a single optional parameter, `boundBy`, inherited by each its subclasses. `boundBy` may either have the value `Device` or `Policy`, indicating that the parameter is to be bound by the XG radio or bound as part of the policy specification, respectively. If the `boundBy` property is not specified, the value defaults to `Policy`. The `Parameter` class is specified as follows:

```
(deftemplate Parameter
      (slot id)
      (slot boundBy)
)
```

Parameters that are defined using a single magnitude and units are subclasses of the `Param` class where `magnitude` is a floating-point number and `unit` is one of the units defined in Section 3.13.

```
(deftemplate Param
      (slot id)
      (slot boundBy)
```

```
        (slot magnitude)
        (slot unit)
)
```

Parameters that represent range values using a minimum and maximum value and units are subclasses of the `ParamRange` class where `max` and `min` are floating point numbers representing the maximum and minimum values of the range, inclusively, and `unit` is one of the units defined in Section 3.13.

```
(deftemplate ParamRange
        (slot id)
        (slot boundBy)
        (slot max)
        (slot min)
        (slot unit)
)
```

While a number of parameters fit into one of the two classes above, we provide a generic construct for ones that do not. Such parameters are subclasses of the `ParamObj` class, with properties defined by the particular subclass.

```
(deftemplate ParamObj
        (slot id)
        (slot boundBy)
)
```

3.4 Processes

Policies may require certain functions that have implementation semantics that are outside the scope of the language but instead can be grounded by an implementation of the function within the radio platform. These functions are specified using `Process` facts, analogous to function prototypes in C. A `Process` fact consists of the following properties:

- `input`–Zero or more parameters required as input to the process.
- `inputOpt` – Zero or more optional input parameters.
- `output`–Zero or more parameters expected as output from the process.
- `xgx`–an optional XG expression that may be used to describe the relationship between the input and output parameters and any limits on them.

```
(deftemplate Process
        (slot id)
 (multislot input)
 (multislot inputOpt)
 (multislot output)
        (slot xgx)
)
```

The parameters used for `input`, `inputOpt`, and `output` must be declared as part of the policy. The parameters are not bound to any values and are not bound by the device. A radio must ground all the processes required by the policy to be able to evaluate whether a policy rule describes a valid opportunity or not. The process types that the policy rule requires are listed in its device description.

3.5 Structure of Authority and Delegation

An authority is an entity that is authorized to establish and enforce policy for a particular jurisdiction. The jurisdiction is defined as a set of frequencies, a set of geographical regions, and a set of time periods. Additionally, an authority may exert control over one or more types of devices.

While an authority establishes the policies for a jurisdiction, another entity may administer and encode the policies. The `PolicyAdministrator` class specifies this entity. Currently the class only names the entity, but may be further developed in future versions of this document. The `PolicyAdministrator` class is defined as follows:

```
(deftemplate PolicyAdministrator
      (slot id)
)
```

An authority may delegate portions of its jurisdiction to another authority. When an authority, X, creates a policy that involves a jurisdiction that it has delegated to another authority, Y, authority X must indicate whether an intersecting policy provided by Y may or may not have precedence over the policy. If X's policy must precede Y's policy, then it indicates that Y cannot enforce a policy that conflicts with X's policy. If Y's policy may precede X's, then Y is allowed to enforce policy that conflicts with X's policy. An example of how to specify these precedence rules can be found in Section 6.6

The description of delegation will be developed further in future versions of this document. The authority fact will be extended to include information to specify delegation information, as well as information assurance information (such as keying material, signatures, and message-digests), and other information describing the authority.

3.6 Frequency Description

This ontology specifies some foundation classes related to frequencies and frequency ranges.

A `FrequencySpecification` can be either a `FrequencyRange` or a `FrequencyGroup`. The `FrequencyRange` class is a subclass of the `ParamRange` class (Section 3.3) so it has a `min` value representing the lower bound of the range, a `max` value representing the upper bound of the range and a `unit` value. The unit value is restricted to the class `FrequencyUnit`. `FrequencyRange` is defined as follows:

```
(deftemplate FrequencyRange
      (slot id)
      (slot boundBy)
      (slot max)
      (slot min)
      (slot unit)
)
```

Multiple FrequencyRanges may be grouped together using a FrequencyGroup. It has a single property, members, which takes one or more FrequencyRanges or FrequencyGroups:

```
(deftemplate FrequencyGroup
      (slot id)
 (multislot members)
)
```

A FrequencyBand is a subclass of FrequencyRange that may specify channel information within the band. A FrequencyBand may optionally specify a channel width for channels in the band. The channelWidth property specifies a Bandwidth class. The FrequencyBand may also optionally specify a channel number, startChannelNum, to start counting channels in that band. FrequencyBand is specified as follows:

```
(deftemplate FrequencyBand
      (slot id)
      (slot boundBy)
      (slot max)
      (slot min)
      (slot unit)
      (slot channelWidth)
      (slot startChannelNum)
)
```

Similarly, Channel is a subclass of FrequencyRange that specifies a specific channel in a band. In addition to specifying the frequency range of the band, it optionally specifies a channel number for the channel. Channel is specified as follows:

```
(deftemplate Channel
      (slot id)
      (slot boundBy)
      (slot max)
      (slot min)
      (slot unit)
      (slot channelNum)
)
```

In addition to specifying classes with frequency ranges, the frequency ontology includes classes that specify single frequencies. The first is the Frequency class that is a subclass of the Param class (Section 3.3). It has a magnitude of the frequency and a unit value that is restricted to be a FrequencyUnit. Frequency is defined as follows:

```
(deftemplate Frequency
      (slot id)
      (slot boundBy)
      (slot magnitude)
      (slot unit)
)
```

The Bandwidth class is similarly defined as a subclass of Param, except it specifies a bandwidth. It is defined as follows:

```
(deftemplate Bandwidth
      (slot id)
      (slot boundBy)
      (slot magnitude)
      (slot unit)
)
```

This ontology may be extended in the future to include additional concepts such as specifying frequencies by a center frequency and bandwidth in addition to the start and stop frequencies above.

3.7 Region Description

This ontology specifies some foundation classes to define geographic areas, locations, and distances.

A `GeographicSpecification` is either a `GeographicArea` or a `GeographicRegion`. A `GeographicArea` is a subclass of `ParamObj`. It represents a named area as it has no properties defining an area. It is intended to be a parent class for more specific area classes, however it may be used to name areas that have well known regulatory meaning. A `GeographicArea` is simply defined as:

```
(deftemplate GeographicArea
      (slot id)
      (slot boundBy)
)
```

A `GeographicRegion` is a group of one or more `GeographicArea`s or `GeographicRegion`s. The included regions and areas are specified using the `includesArea` property as specified below:

```
(deftemplate GeographicRegion
      (slot id)
 (multislot includesArea)
)
```

Currently two subclasses of `GeographicArea` are included in the ontology, a `CylindricalArea` and `SphericalArea`. A `CylindricalArea` defines a cylinder of space specified by the properties: `centerAt`–the geographic coordinate of center of the cylinder, `heightOf` – the height of the cylinder, and `radiusOf`–the radius of the cylinder. These properties specify a `GeographicCoordinate` and a `Radius`, respectively. `CylindricalArea` is specified as follows:

```
(deftemplate CylindricalArea
      (slot id)
      (slot boundBy)
      (slot centerAt)
      (slot heightOf)
      (slot radiusOf)
)
```

A `SphericalArea` defines a sphere in space. As with `CylindricalArea` it specifies the space with the properties `centerAt` and `radiusOf`. Additionally it has the optional property of `altitudeOf` that specifies the altitude of the sphere's center. `altitudeOf` specifies an `Altitude` parameter. `SphericalArea` is specified as follows:

```
(deftemplate SphericalArea
      (slot id)
      (slot boundBy)
      (slot centerAt)
      (slot radiusOf)
      (slot altitudeOf)
)
```

Future versions of this document may include additional geographic areas such as a polygonal area that defines a polygon as an ordered list of geographic coordinates.

A geographic coordinate is specified with the GeographicCoordinate class. It has two properties, longitude and latitude, that specify the longitude and latitude of the coordinate, respectively. The properties longitude and latitude have values specified in decimal format with north and east being represented by positive numbers and south and west using negative numbers. An example is provided in Section 6.1.
GeographicCoordinate is specified as follows:

```
(deftemplate GeographicCoordinate
      (slot id)
      (slot latitude)
      (slot longitude)
)
```

In addition to GeographicAreas, this ontology provides several parameters related to distances. Distance is a subclass of the Param class (Section 3.3). It has a magnitude of the distance and a unit value that is restricted to be a DistanceUnit. Distance is defined as follows:

```
(deftemplate Distance
      (slot id)
      (slot boundBy)
      (slot magnitude)
      (slot unit)
)
```

Currently three subclasses of Distance are defined: Radius, Height, and Altitude. These subclasses are defined the same as Distance and represent a radius, a height, and an altitude, respectively.

3.8 Time Description

The time ontology describes three forms of time: time intervals, time instants, and time durations.
This ontology depends on the DateTime class that specifies a single date and time. DateTime is specified as follows:

```
(deftemplate DateTime
      (slot id)
      (slot tdyear)
      (slot tdmonth)
      (slot tdday)
      (slot tdhour)
      (slot tdminute)
      (slot tdsecond)
      (slot tdusecond)
)
```

Each property takes an integer value that represents a calendar year, calendar month (1 = January ... 12 = December), day of the month, hour (24 hour time), minute, second, and microsecond respectively.

A TimeInstant is a moment in time defined by a date and a time. TimeInstant is a subclass of both DateTime and ParamObj (Section 3.3) and inherits all properties from both.

A TimeInterval is a ParamObj that defines a range of time that occurs between instants of time. A TimeInterval has two properties: starttime that specifies a DateTime that

marks the beginning of the time range and `endtime` that specifies a `DateTime` that marks the end of the time range. `TimeInterval` is defined as:

```
(deftemplate TimeInterval
      (slot id)
      (slot boundBy)
      (slot starttime)
      (slot endtime)
)
```

Finally, a parameter `TimeDuration` defines an elapsed amount of time. `TimeDuration` is a subclass of the `Param` class (Section 3.3). It has a `magnitude` of the elapsed time and a `unit` value that is restricted to be a `TimeUnit`. `TimeDuration` is defined as follows:

```
(deftemplate TimeDuration
      (slot id)
      (slot boundBy)
      (slot magnitude)
      (slot unit)
)
```

This ontology may be extended further to better support a formal temporal calculus, express time zones, and support ISO standard representation of dates and times [ISO8601]. It may also be extended to support policies that specify additional time statements such as “on Mondays” or “on the last day of a month”. This ontology may be partially replaced with the OWL Time ontology [DAMLOT] in the future.

3.9 Device Description and Capabilities

The device ontology is used to define a device that can be specified in a device description, device capabilities, and parameters that relate to the device and its measurements.

A device is defined by both its device type and capabilities. `DeviceTyp` defines the type of XG device. Device types are individuals of the class `DeviceTyp`, which is defined with no properties as:

```
(deftemplate DeviceTyp
      (slot id)
)
```

A device’s capabilities are defined in the class `DeviceCap`. `DeviceCap` has three parameters that are used to define the device capabilities. `hasPolicyDefinedParams` is one or more parameters that an XG device must be able to understand and bind in order to process a set of policy rules. `hasPolicyDefinedBehaviors` similarly lists one or more processes that an XG device must be able to ground in order to process the set of policy rules. `hasDeviceCapabilities` states one or more capabilities that the device must have to process the set of policies. A list of capabilities will be defined in future versions of this document. `DeviceCap` is defined as:

```
(deftemplate DeviceCap
      (slot id)
 (multislot hasDeviceCapabilities)
 (multislot hasPolicyDefinedParams)
 (multislot hasPolicyDefinedBehaviors)
)
```

The device ontology includes regulatable parameters and processes that are related to the device. The parameters and processes currently in the ontology are representative of

some of the parameters and processes that will be part of the ontology. They are not a complete set and will likely be expanded in future versions of this document.

The parameters currently included in the ontology are:

- `NumChannels` – A class that defines the number of channels a device may use as part of a policy. This class is a subclass of `Count` defined in Section 3.12.
- `ChannelNum` – A class that defines a specific channel number to use. `ChannelNum` is a subclass of `Param` (Section 3.3) with a `magnitude` of the channel number and a `unit` value restricted to `NoUnit`.
- `SignalType` – A class defining the type of a signal (e.g., TV signals such as NTSC, PAL, DTV). `SignalType` is a subclass of `ParamObj` (Section 3.3) with no further restrictions.
- `Leakage` – A class that specifies the amount of signal that may leak outside of the transmission band. `Leakage` is a subclass of `Param` with a `magnitude` of the percent leakage and a `unit` value restricted to `Percent`.

The processes currently defined are:

- `Sensing` – A class that specifies a process that retrieves data from sensing. `Sensing` is a subclass of `Process` (Section 3.4).

3.10 Environment and Device State

This ontology is a placeholder to describe the state of the environment and the XG radio. Only those parameters related to environment and device state that are known to, and can be regulated by, policy are of interest. An example is a low spectrum usage area such as a rural area. Other examples of parameters that may be included in this ontology include: current weather (e.g., precipitation, wind-speed, temperature), spectral maps as measured by the XG radio, and neighbors of the radio within a network.

This ontology will be developed in future versions of this document.

3.11 System-Dependent Extensions

The XG policy language must support system-specific extensions that will be used to define system policy. This ontology will include any classes necessary to support the definition of system-specific policy. Any parameters that are not directly of interest to regulatory bodies, but are of concern to system performance, such as battery management are also appropriate for inclusion here. For example, in the case of tactical policy, it will be useful to include a description of the military structure in this ontology.
This ontology will be developed in future versions of this document.

3.12 Physical Quantities

This ontology describes physical quantities with scope more general than their use within spectrum policy. Specific parameters based on physical quantities that have a richer structure in the spectrum policy context (e.g. frequency band classifications) are treated in more detail in the respective XG policy language ontologies (e.g. frequency, time, and

region ontologies). The quantities that are currently in this ontology are only a representative set—a more extensive set of physical quantities (based on the SI system of units, for example) will be included in future versions of this document.

Some quantities currently defined are:

- Field Strength – The class `FieldStrength` defines the field strength quantity. `FieldStrength` is a subclass of `Param` (Section 3.3) with `unit` value restricted to `FieldStrengthUnit`.
- Power Spectral Density – The class `PowerSpectralDensity` defines the power spectral density quantity. `PowerSpectralDensity` is a subclass of `Param` with `unit` value restricted to `PSDUnit`.
- Power – The class `Power` defines the power quantity. `Power` is a subclass of `Param` with `unit` value restricted to `PowerUnit`.
- Count – The class `Count` defines a counted quantity. `Count` is a subclass of `Param` with `unit` value restricted to `NoUnit`.

3.13 Units

Parameter units are included as part of the XG policy language to support defining parameter values. A `Unit` class is defined and is subclassed for each family of units. The ontology currently contains the following units—a more complete set of units (e.g. based on the SI system, compound units) will be included in future versions of this document.

Unit Class	***Instances***
NoUnit	None
FrequencyUnit	Hz kHz MHz GHz THz
FieldStrengthUnit	uVperm mVperm Vperm dBuVperm
DistanceUnit	mm cm m km mi Nmi
DecibelUnit	dB
PowerUnit	uW mW W kW MW dBW dBm
PercentUnit	Percent
TimeUnit	nsec usec msec sec minute hr day wk mon yr
PSDUnit	nWperHz, uWperHz, mWperHz, WperHz, dBmperHz

4 Policy Processing

In this section we describe the rules for processing the policy facts defined in Section 3; these rules implement a part of the concept of operations described in Section 3.1.2. We focus on checking policy conformance of a proposed instance (that is, the function of the policy conformance reasoner in Figure 81), as opposed to techniques that search for policy-conformant solution instances that are suitable for a particular radio. Conformance checking takes as input selector, opportunity and usage constraint instances, and determines if they conform to policy. If only a subset of the instances is provided (for example only a selector instance), all policies that match the provided subset are returned. Note that a system strategy reasoner can make use of this capability provided by the policy conformance reasoner to search for solution instances in a system-dependent fashion. For example, a set of usage descriptions provides a set of constraints that can be solved using a constraint solver.

Figure 84 illustrates the conformance checking process, a three-step process that takes as inputs the policy instance and related facts, a selector instance, an opportunity instance, and a usage instance. Each instance contains parameter bindings (and process groundings) required to determine the validity of the corresponding description. Note that while we present this processing as sequential steps, an implementation of the policy conformance reasoner may process the selector, opportunity, and usage descriptions in parallel and combine the results at the end. The policy rules in Section 4.3 below demonstrate this.

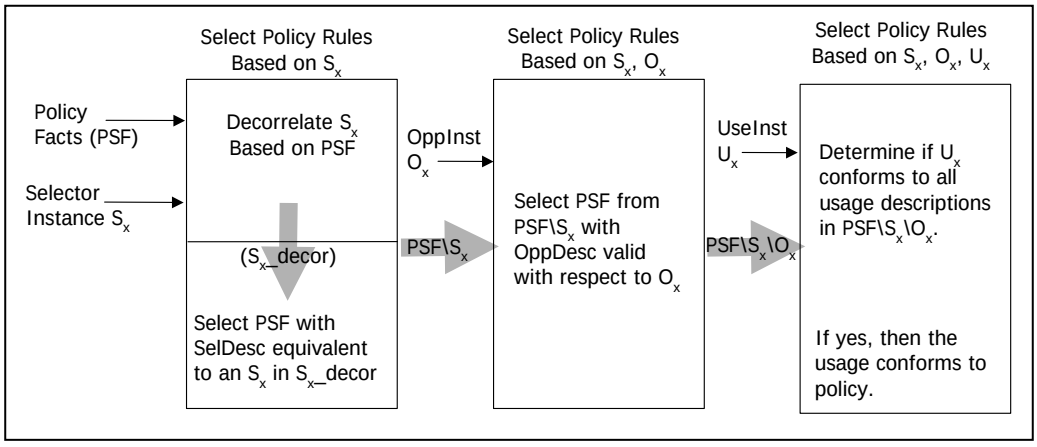


Figure 84: Policy Conformance Checking Process

In the first step, the conformance checking process takes the policy facts (PSF) and a selector instance (S_x). Given the selector instance and a selector description from PSF, we can assert that the two are disjoint, equivalent, or correlated. If S_x is correlated with a selector description, it must be decorrelated against that description so that the decorrelated instances are either disjoint or equivalent to the description. For further discussion, we will assume that S_x is already decorrelated with respect to the selector descriptions in PSF. The policy rules in PSF with selector descriptions that are equivalent to S_x are *selected*–we denote this set of selected policy rules by $PSF\backslash S_x$, (where "\" is the notation for

"conditioned upon"). The system strategy reasoner may use this subset to create suitable opportunity instances.

In the second step, the opportunity instance (O_x), if provided, is compared with all opportunity descriptions within PSF\$S_x$. If the instance satisfies an opportunity description for a selected policy rule, then depending on the value of the deny property of the policy rule, it is asserted that the instance is either valid or invalid with respect to the policy rule. If the validity of the instance is not contradicted by a rule of equal or greater precedence (after performing any additional meta-policy processing), then the usage constraints imposed by the rule *applies*. If the instance does not satisfy the description, then the instance is not germane, and the policy rule does not apply. We denote the subset of rules in PSF\$S_x$ that apply by PSF\$S_x$\$O_x$.

We note that the usage constraint descriptions in PSF\$S_x$\$O_x$ are essentially the constraint expressions that limit usage by the radio, after taking into account the radio configuration, the situation of the radio, and the intended mode of operation. The constraints selection process is coarsely governed by the selector instance, and more finely by the opportunity descriptions that match.

In the final step, the usage instance (U_x), if provided, is considered. For each usage constraint description U_i of a policy rule in PSF\$S_x$\$O_x$, we can assert either that U_x is or is not a solution of U_i, depending on whether or not the instance satisfies the description. If, for all usage constraints in PSF\$S_x$\$O_x$, U_x is a solution of the usage descriptions, then we can assert that the usage conforms to policy. The radio is then permitted to use spectrum as defined by the set of instances S_x, O_x, and U_x.

We note that the usage constraints apply conjunctively unless policy rules are organized into disjunctive groups using meta-policy. However, we do not describe how to handle processing of disjunctive groups of policies in this version of the RFC, but future versions may include rules for processing disjunctions.

We also note that there is a *condition-implies-action* idiom that is implicit in the structure of the policy rules (and hence the use of the term *rule*). Matching the (selector and) opportunity description is equivalent to matching the *condition* of a rule, which in turn implies that the corresponding usage constraint description will apply, equivalent to the *action* part of the rule. This implicit idiom is enforced explicitly using rules in the following subsection.

The processing of policy rules as described above (even taking into account precedence relations described later) to test for conformance can be done efficiently[17]. Furthermore, the processing of policy rules can be done in parallel to a large extent. A rigorous characterization of the expressiveness of the policy language and the complexity of processing meta-policies (as we develop them further) is a subject of future work.

[17] In polynomial time in the number of policies (assuming the time to invoke processes grounded by the radio is bounded from above by a constant)

The following sections define the processing rules needed to implement this processing.

4.1 Shorthand Notation for Rules

In this section, we introduce a shorthand notation for describing rules as we previously defined a shorthand notation for facts. Like the facts shorthand, the rules shorthand is based on an existing rule-based knowledge representation environment [CLIPS].

A rule is defined as follows:

```
(defrule <rule-name>
  [(declare (salience <salience>))]
  <condition>
  =>
  <action>)
```

Salience is an optional declaration that provides rule priority. If multiple rules have a condition that is satisfied, the rules with a higher salience will fire first. If all the elements in the condition are satisfied then the expression in the action is executed.

The condition is a set of one or more condition elements. All the elements must be satisfied for the condition to be satisfied. A condition element is one of:

- The logical operators `and`, `or`, and `not`. Additionally, the operator `exists`. `Exists` returns true if the fact provided is asserted, false otherwise.
- A predicate. A predicate must be called using the `test` element. For example if we have a predicated named `doSomething`, then we can use it in the condition as follows

  ```
  (test (doSomething ?arg1 ?arg2))
  ```
- A template pattern. A condition may require information from a fact that has been asserted. This information is obtained by using a template pattern. The pattern states the known part of a fact and provides variables for unknown values that are to be obtained. For example, if we want the name of a policy rule that has `TRUE` for the value of the `deny` slot, we would use the following template:

  ```
  (PolicyRule (id ?pid) (deny TRUE))
  ```
- An assignment pattern. An assignment pattern assigns a template pattern to a variable. This assigns the fact itself to the variable, not the name, of the fact. An example of an assignment pattern is:

  ```
  ?f1 <- (PolicyRule (id ?pid) (deny TRUE))
  ```

Variables are symbols prefixed with a `?`, such as `?pid` in the example above.

The action is an expression similar to the XG expression described in Section 3.1.2, however it is not limited to a single expression and it is not enclosed in quotation marks.

4.2 Instance Facts

The XG radio provides a set of parameter bindings as input to the policy processing. The bindings take the form of a selector instance, an opportunity instance and a usage instance. Each instance is represented by one or more facts described in this section.

A selector instance is defined in the `SelInst` class. The `SelInst` class has four slots, `freqDesc`, `regnDesc`, `timeDesc`, and `devcDesc`, which are defined as the like named slots in the selector description described in Section 3.2.1.

```
(deftemplate SelInst
       (slot id)
       (slot freqDesc)
       (slot regnDesc)
       (slot timeDesc)
       (slot devcDesc)
)
```

An opportunity instance is defined in the `OppInst` class. The `instBind` slot provides the names of one or more parameter bindings as described below.

```
(deftemplate OppInst
       (slot id)
  (multislot instBind)
)
```

A usage instance is defined in the `UseInst` class. As with the opportunity instance, the `instBind` slot provides the names of one or more parameter bindings. Additionally, the usage instance contains the selector instance and opportunity instance associated with the usage instance in the `selInst` and `oppInst` slots, respectively.

```
(deftemplate UseInst
       (slot id)
       (slot selInst)
       (slot oppInst)
  (multislot instBind)
)
```

The parameter bindings are stated using the `InstBinding` class. This class associates a parameter name specified in the policy to a parameter value provided by the radio. The slot `paramName` contains a parameter specified in the policy as being bound by the device. The `paramValue` slot contains a parameter with a value that has been bound by the device.

```
(deftemplate InstBinding
       (slot id)
       (slot paramName)
       (slot paramValue)
)
```

The rules also depend on a variety of other facts that are internal to the rule processing. They are used to indicate the result of some processing so that result is available to subsequent rules. These facts will be explained as they are used to define rules below.

4.3 Rules for Policy Processing

This section presents a set of rules to perform policy conformance checking described above. These rules are intended to illustrate the conformance checking process. A policy conformance reasoner may use a different implementation. We also note that a full implementation of a conformance reasoner will require additional rules such as rules to check completeness of the policy, rules to support the grounding of the predicates in the actions, and other implementation specific rules.

We start by looking at the rules for determining if an instance (selector, opportunity, and usage) conforms to policy. We then look at each type of instance individually and how to determine if it satisfies policy. Finally, we will look at some rules for processing meta-policies.

4.3.1 Policy Conformance

These rules test whether or not an instance conforms to policy. Barring any meta-policy rules to the contrary, if an instance does not satisfy the usage description of any policy rule for which it selects and is valid, then it does not conform to policy. We can state a rule that implements this as:

Rule in Shorthand Notation	Remarks

(defrule
  instance-I-conform-to-policy
(declare (salience –100))
(UseInst (id ?ui)
  (selInst ?si) (oppInst ?oi))
(PolicyRule (id ?pid)
  (useDesc ?ud))
(SelectPolicyRule
  (polRule ?pid) (selInst ?si))
(ValidOppPerPolRule
  (polRule ?pid) (oppInst ?oi))
(UseNotSatisfied
  (inst ?ui) (desc ?ud))
=>
(assert
  (InstanceDoesntConformToPol
  (useInst ?ui))))
	This must be done after conformance to selector, opportunity, and usage descriptions are tested.
	Match a usage instance template.
	Match a policy rule template.
	Check if the policy rule has been asserted to be selected by the instance.
	Check if the opportunity instance has been asserted to be valid per the policy rule.
	Check if the usage instance has been asserted to not conform to the usage description in the policy rule.
	Assert that the usage instance does not conform to policy.

An instance conforms to policy if its usage satisfies all the policy rules for which it is selected and valid. The following rule fires after the test for non-conformance and verifies that non-conformance has not been asserted.

Rule in Shorthand Notation	Remarks

(defrule
  instance-conforms-to-policy
(declare (salience –110))
(UseInst (id ?ui)
  (selInst ?si) (oppInst ?oi))
(PolicyRule (id ?pid))
(SelectPolicyRule
  (polRule ?pid) (selInst ?si))
(ValidOppPerPolRule
  (polRule ?pid) (oppInst ?oi))
(not (exists
  (InstanceDoesntConformToPol
  (useInst ?ui))))
=>
(assert (InstanceConformsToPol
  (useInst ?ui))))
	This must be done after non-conformance is tested.
	Match a usage instance template.
	Match a policy rule template.
	Check if the policy rule has been asserted to be selected by the instance.
	Check if the opportunity instance has been asserted to be valid per the policy rule.
	Check if the usage instance has been determined to not conform to policy.
	Assert that the usage instance conforms to policy.

We will now develop the rules to test if a policy rule is selected and valid and if a usage constraint is satisfied.

4.3.2 Selecting Policy Rules

This section describes the rules to select policy rules based on a supplied selector instance. A policy rule is selected by a selector instance if the selector instance is equivalent to the selector description contained in the policy rule. To illustrate:

Rule in Shorthand Notation	Remarks

`(defrule select-policyrules`
` (declare (salience -5))`
` (SelInst (id ?si))`
` (PolicyRule (id ?pid)`
`  (selDesc ?sd))`
` (SelEquivalent`
`  (inst ?si) (desc ?sd))`
` =>`
` (assert (SelectPolicyRule`
`  (polRule ?pid) (selInst ?si)))`
	
	Match a selector instance template.
	Match a policy rule template.
	
	Match a template indicating the selector instance and description are equivalent (set above).
	
	Assert that the policy rule is selected per the selector instance.

In order to determine if a selector instance and description are equivalent, we need to check if the authority in the selector description has authority over the space stated in the selector instance and that each sub description (frequency, region, time, and device) in the instance intersects with the corresponding sub description in the selector description.

For each of these comparisons, we can assert zero, one, or two facts. If the sub description from the selector instance and the selector description intersect, then we assert that the two descriptions are correlated. If they are exact matches then we also assert that they are equivalent. Similarly when testing jurisdiction, we assert that the jurisdiction is correlated if the authority's jurisdiction intersects the instance space, and we additionally assert that the authority has jurisdiction over the instance space if the space is completely contained within the authority's jurisdiction. We can state a rule to test the jurisdiction and intersection of each sub descriptor in each selector instance/description pair as follows:

Rule in Shorthand Notation	Remarks
(defrule compare-SelInst (declare (salience 0)) (SelInst (id ?si) (freqDesc ?fi) (regnDesc ?ri) (timeDesc ?ti) (devcDesc ?di))	Match a selector instance template.
(SelDesc (id ?sd) (authDesc ?ad) (freqDesc ?fd) (regnDesc ?rd) (timeDesc ?td) (devcDesc ?dd)) =>	Match a selector description template.
(compareJurisdiction ?ad ?fi ?ri ?ti ?di)	Test if the authority in the description has authority over the frequency, region, time, and device presented in the selector instance. If the authority has complete jurisdiction over the space, then assert AuthHasJurisdictionOver and AuthCorrelatedJurisdiction. If it has overlapping, but not complete jurisdiction, assert only the latter.
(compareDesc ?fd ?fi)	Test if the frequency instance and descriptor intersect. Assert DescEquivalent and DescCorrelated if they are the same. If they overlap, but are not the same, assert only the latter.
(compareDesc ?rd ?ri)	Test if the region instance and descriptor intersect, assert as described above.
(compareDesc ?td ?ti)	Test if the time instance and descriptor intersect, assert as described above.
(compareDesc ?dd ?di)	Test if the device instance and descriptor intersect, assert as described above.

The above rule compares each of the components of the selector instance and description and asserts any intersection. The following rule uses these assertions to determine if the selector instance is equivalent to the selector description. They are equivalent if the authority in the description has jurisdiction over the frequency, region, time, and device in the instance, and the instance's frequency, region, time, and device descriptions are equivalent to those in the selector description.

Rule in Shorthand Notation	Remarks
(defrule equivalent-SelInst (declare (salience 0)) (SelInst (id ?si) (freqDesc ?fi) (regnDesc ?ri) (timeDesc ?ti) (devcDesc ?di))	Match a selector instance template.
(SelDesc (id ?sd) (authDesc ?ad) (freqDesc ?fd) (regnDesc ?rd) (timeDesc ?td) (devcDesc ?dd))	Match a selector description template.
(not (exists (SelEquivalent (inst ?si) (desc ?sd))))	Check that this selector has not been processed (it may be asserted by selDecorrelate)
(AuthHasJurisdictionOver (authDesc ?ad) (freqDesc ?fi) (regnDesc ?ri) (timeDesc ?ti) (devcDesc ?di))	Check that the authority has jurisdiction over the instance.

`(DescEquivalent (desc ?fd) (inst ?fi))`	Check if the frequency instance and descriptor have been asserted to be equivalent.
`(DescEquivalent (desc ?rd) (inst ?ri))`	Check if the frequency instance and descriptor have been asserted to be equivalent.
`(DescEquivalent (desc ?td) (inst ?ti))`	Check if the frequency instance and descriptor have been asserted to be equivalent.
`(DescEquivalent (desc ?dd) (inst ?di))`	Check if the frequency instance and descriptor have been asserted to be equivalent.
`=>`	
`(assert (SelEquivalent (inst ?si) (desc ?sd)))`	Assert that the selector instance and description are equivalent.

If a selector instance isn't equivalent to any selector descriptions, it may still be correlated with one. A selector instance is correlated with a selector description if all the components of the instance and description are correlated.

A selector instance that is correlated with a selector description must be partitioned into a set of selector instances that are either equivalent to the selector description or disjoint with it. For example, decorrelation is required if a system needs to use spectrum that spans multiple bands, and the policy is organized such that each band must be considered separately to determine the policies that apply within that band. The predicate `selDecorrelate` performs the partitioning and assert equivalence for any partition that is equivalent. Decorrelation is illustrated in Figure 85.

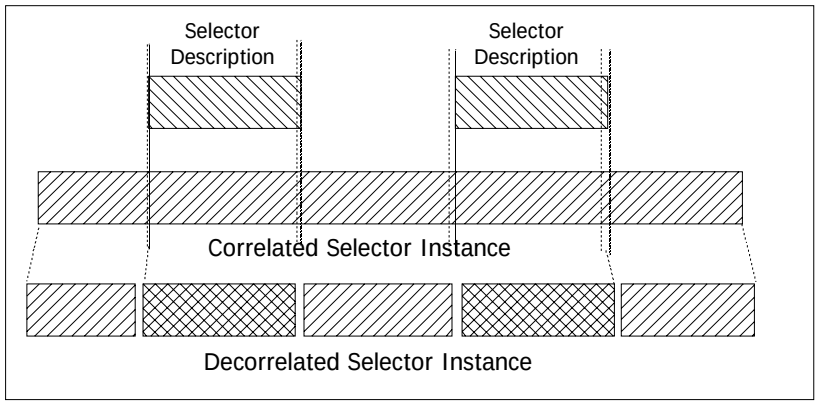


Figure 85: Decorrelation of a Selector Instance

This rule tests for correlation and decorrelates the instance, if necessary:

Rule in Shorthand Notation	Remarks
(defrule correlated-SelInst (declare (salience -5))	Must process these after equivalence has been determined.
(SelInst (id ?si) (freqDesc ?fi) (regnDesc ?ri) (timeDesc ?ti) (devcDesc ?di))	Match a selector instance template.
(SelDesc (id ?sd) (authDesc ?ad) (freqDesc ?fd) (regnDesc ?rd) (timeDesc ?td) (devcDesc ?dd))	Match a selector description template.
(not (exists (SelEquivalent (inst ?si) (desc ?sd))))	Check that this selector has not been determined to be equivalent. An equivalent selector will also be correlated, but we don't need to decorrelate it.
(AuthCorrelatedJurisdiction (authDesc ?ad) (freqDesc ?fi) (regnDesc ?ri) (timeDesc ?ti) (devcDesc ?di))	Check that the authority has overlapping jurisdiction with the instance.
(DescCorrelated (desc ?fd) (inst ?fi))	Check for an assertion that the frequency instance and descriptor are correlated.
(DescCorrelated (desc ?rd) (inst ?ri))	Check for an assertion that the region instance and descriptor are correlated.
(DescCorrelated (desc ?td) (inst ?ti))	Check for an assertion that the time instance and descriptor are correlated.
(DescCorrelated (desc ?dd) (inst ?di))	Check for an assertion that the device instance and descriptor are correlated.
=> (selDecorrelate ?si ?sd))	A function that decorrelates the selector instance with respect to the selector description. The function asserts SelEquivalent for any equivalent instances that result from the decorrelation.

4.3.3 Satisfying Opportunity Descriptions

This section describes the rules for determining if an opportunity instance satisfies an opportunity description and whether it is valid or invalid with respect to a policy rule. First we test each opportunity instance/description pair to see if the parameter bindings provided by the instance satisfy the XG expression provided in the description. Note that although it is not illustrated here, only the opportunity descriptions included in selected policy rules need to be examined. If the instance satisfies the description, we assert that fact.

Rule in Shorthand Notation	Remarks
(defrule opportunity-satisfied-rule (declare (salience 0))	
(SelInst (id ?si))	Match a selector instance template.
(OppInst (id ?oi))	Match an opportunity instance template.
=> (oppSatisfied ?od ?oi))	Test if the XG expression contained in the opportunity description is satisfied by the values bound in the instance. If it is satisfied, then assert OppSatisfied.

If an opportunity instance satisfies a description, it may represent either a valid or invalid opportunity depending on the value of the `deny` slot in a policy rule that contains that opportunity description. We create two rules. The first asserts that the opportunity is valid if `deny` is `FALSE` and the second asserts that the opportunity is invalid if `deny` is `TRUE`.

Rule in Shorthand Notation	Remarks

`(defrule valid-opp-per-policy-rule`
`(declare (salience 0))`
`(OppInst (id ?oi))`
`(PolicyRule (id ?pid)`
`(oppDesc ?od) (deny FALSE))`
`(OppSatisfied`
`(desc ?od) (inst ?oi))`
`=>`
`(assert (ValidOppPerPolRule`
`(polRule ?pid) (oppInst ?oi)))`
	Match an opportunity instance template
	Match a policy rule template. Deny=FALSE indicates the rule represents a valid opportunity.
	Check if the opportunity has been determined to be satisfied.
	
	Assert that this opportunity is valid with respect to this policy rule.

Rule in Shorthand Notation	Remarks

`(defrule`
`invalid-opp-per-policy-rule`
`(declare (salience 0))`
`(OppInst (id ?oi))`
`(PolicyRule (id ?pid)`
`(oppDesc ?od) (deny TRUE))`
`(OppSatisfied`
`(desc ?od) (inst ?oi))`
`=>`
`(assert (InvalidOppPerPolRule`
`(polRule ?pid) (oppInst ?oi)))`
	Match an opportunity instance template.
	Match a policy rule template. Deny=TRUE indicates the rule represents an invalid opportunity.
	Check if the opportunity has been determined to be satisfied.
	
	Assert that this opportunity is invalid with respect to this policy rule.

Note that the opportunity may now be asserted to be valid and invalid with respect to different policy rules. This conflict must be resolved by rules for processing the precedence meta-policy described later.

4.3.4 Satisfying Usage Constraint Descriptions

This section describes the rule for determining if a usage instance satisfies a usage description. To accomplish this, we test each usage instance/description pair to see if the parameter bindings provided by the instance satisfy the XG expression provided in the description. Note that although it is not illustrated here, only the usage descriptions included in selected policy rules that conclude the opportunity instance is valid need to be examined. If the instance satisfies the description, we assert that fact.

Rule in Shorthand Notation	Remarks
(defrule usage-conformance (declare (salience 0)) (UseInst (id ?ui)) (UseDesc (id ?ud)) => (matchUsage ?ud ?ui))	 Match a usage instance template. Match a usage description template. Test if the XG expression contained in the usage description is satisfied by the values bound in the instance. Assert UseSatisfied if it is satisfied, UseNotSatisfied, otherwise.

4.3.5 Meta Policy

Meta-policy rules modify the processing rules described above by describing relationships between policy rules. This version of this document describes only rules for processing precedence meta-policies. Rules for processing other meta-policies as well will be developed further in future revisions of this document. These meta-policy processing rules will address issues such as policy rule precedence under delegation of authority, precedence of derived policy rules, re-computation of derived rules under addition and revocation of policy rules, and disjunctive groups of policy rules.

4.3.5.1 Precedence

Precedence is used to determine which policy rules apply if multiple policy rules conflict. If a highest precedence policy rule whose opportunity description is satisfied by an instance concludes that the instance is invalid, then none of the other policy rules of equal or lower precedence apply. Otherwise (i.e., all rules of the highest precedence conclude the instance is valid), all policy rules that conclude the instance is valid (including policy rules with lower precedence) do apply.

The following two rules implement precedence. The first retracts an assertion that an opportunity instance is invalid if the policy rule is strictly preceded by another policy rule that asserts the instance is valid.

Rule in Shorthand Notation	Remarks
(defrule remove-invalid-preceded-by-valid (declare (salience –10)) (OppInst (id ?oi)) (PolicyRule (id ?pid1)) (PolicyRule (id ?pid2)) (ValidOppPerPolRule (polRule ?pid1) (oppInst ?oi)) ?f1 <- (InvalidOppPerPolRule (polRule ?pid2) (oppInst ?oi)) (PolPrecedes (left ?pid1) (right ?pid2)) (not (exists (PolPrecedes (left ?pid2) (right ?pid1)))) => (retract ?f1))	This must happen after all valid/invalid assertions are made. Match an opportunity instance template. Match a policy rule template. Match another policy rule template. Check for an assertion that the first policy rule is valid with respect to the instance. Check for an assertion that the second policy rule invalidates the instance and assign this fact to f1. Check for an assertion that the first policy rule precedes the second. Check that the second policy rule does not also precede the first (e.g., they have equal precedence). Retract the fact f1 – the second policy rule is invalid with respect to the instance.

The second retracts a valid assertion if the policy rule is preceded by another rule that asserts the opportunity instance is invalid.

Rule in Shorthand Notation	Remarks
(defrule remove-valid-preceded-by-invalid (declare (salience –20)) (OppInst (id ?oi)) (PolicyRule (id ?pid1)) (PolicyRule (id ?pid2)) (InvalidOppPerPolRule (polRule ?pid1) (oppInst ?oi)) ?f1 <- (ValidOppPerPolRule (polRule ?pid2) (oppInst ?oi)) (PolPrecedes (left ?pid1) (right ?pid2)) => (retract ?f1))	This must occur after all policy rules that assert an instance is invalid and are preceded are removed first. Match an opportunity instance template Match a policy rule template. Match another policy rule template. Check for an assertion that the first policy rule is invalid with respect to the instance. Check for an assertion that the second policy rule is valid with respect to the instance and assign this fact to f1. Check for an assertion that the first policy rule precedes the second. Retract the fact f1 – the second policy rule is valid with respect to the instance.

Both rules depend on the `PolPrecedes` fact. In some instances, this fact will be stated explicitly in the policy, however, other cases will require the fact to be asserted. First, precedence is transitive. In order to ensure the above rule works, we may compute the transitive closure of the precedences. Second, precedence may be expressed as precedences between groups of policies. If group G_y precedes group G_z then all the policy rules in G_y precede the rules in G_z.

5 Ontology Extension

The XG policy language is expected to evolve over time as regulatory needs and device capabilities change. Perhaps the evolution of XGPL will eventually take place within the auspices of a standards body. Nonetheless, it may be necessary to extend the keywords supported by the language outside of the standards process. For example, the expression of system-specific policy may require parameters and processes that are outside of the regulatory domain and may not be included in the language. While it is not difficult to extend the policy language, an XG radio must also be able to understand the new keywords in order to use the policies that make use of the extensions.
The XG policy language can be readily extended by adding process and parameter keywords. Furthermore, the use of OWL as the underlying representation enables the XGPL ontologies to be extended using OWL statements, and additional ontologies to be defined (or ones defined elsewhere to be imported) for use within this framework. In this section we will discuss how additional processes and parameters can be added, and how the XGPL ontologies can be extended significantly.

5.1 Adding Keywords for Processes and Parameters

Adding a new process is just a matter of defining a process fact with a new process type. For example, we can define a process named `joinAreas`, which takes two geographic areas as arguments and returns a geographic area that is the union of those two areas. This process could be defined as follows:

```
(Process (id joinAreas)
     (input GeographicArea1 GeographicArea2)
     (output GeographicArea3)
)
```

A policy rule that uses this process must indicate in its device description that the XG radio must support the `joinAreas` process in order to use the rule. It follows that an XG radio that wishes to operate under any policy involving `joinAreas` must understand what it means (that is, implement the process and provide a grounding for it).

Similarly, we can add a parameter keyword to the language provided the parameter type is already defined by the ontology.

5.2 Adding New Parameter Types

We can also add new parameter *types* to the language. Although we will continue to use the shorthand notation to illustrate this, we note that adding new types requires more familiarity with OWL features than we have required so far. In future revisions, we may enhance the syntax of the surface notation to make this more transparent to the policy writer by, for example, the use of a native parameter type declaration within the XGPL ontologies.

If the new parameter type to be added is either a scalar quantity that is specified using a magnitude and a unit or a quantity with a continuous range of scalar values specified using the minimum and maximum of the range and a unit, then we can add the new parameter by subclassing `Param` or `ParamRange` respectively. New parameters that do not

conform to the structure of `Param` or `ParamRange` can extend the more general `ParamObj` class or its subclasses.
For example, we can create a new parameter, Bandwidth, as follows:

```
(owl:Class (id Bandwidth)
  (rdfs:subClassOf Param))
```

This new class will inherit the slots `magnitude` and `units` from `Param`. It is likely that we will specify additional OWL statements that restrict the units to frequency units (not shown here for brevity). A corresponding deftemplate and symbol mappings may also be defined for use with the shorthand notation, either manually or using an automated tool.

```
(deftemplate Bandwidth
  (slot magnitude)
  (slot units))
```

5.3 Extending the XGPL Ontologies

In the previous section we discussed how new parameter types can be added to the language by using OWL statements. OWL statements can be used to extend the language in significant ways over and above adding new types of parameters. Suppose we wish to extend the language in order to specify policy within a military context. One useful extension in this case would be to incorporate knowledge about military structure.

For example, consider the following knowledge about the military structure. The soldier is a fundamental entity within the structure. There are typically 9-10 soldiers per squad, 2-4 squads per platoon, 3-5 platoons per company, and 4-6 companies per battalion, etc. We can begin to represent that knowledge by using OWL statements in the shorthand notation as follows:

No.	Encoded Policy in Shorthand Notation	Remarks
1	`(owl:Class (id Soldier))`	Create a new class identified by Soldier.
2	`(owl:ObjectProperty (id hasSoldier))`	Create an object property called hasSoldier, indicating the property takes only individuals (objects not data-types) as values.
3	`(owl:Class (id Squad) (rdfs:subClassOf R1 R2))`	Create a class called Squad, which is described further by two restrictions R1 and R2 below. which limits 10 soldiers to a squad.
4	`(owl:Restriction (id R1) (owl:onProperty hasSoldier) (owl:maxCardinality 10))`	The hasSoldier Property can take at most 10 values.
5	`(owl:Restriction (id R2) (owl:onProperty hasSoldier) (owl:allValuesFrom Soldier))`	The hasSoldier can take only values that are individuals of class Soldier.

In the shorthand notation, this can be represented using the following (which hides a lot of the background knowledge):

```
(deftemplate Soldier (slot id))
(deftemplate Squad (slot id)) (multislot hasSoldier))
```

Given these extensions, we can now go on to further define Unit Ids for individual squads and specify policy rules that enable certain squads to use certain frequency allocations at certain times with XG radios. Clearly, in order to use these extensions, the radio platform must ground its semantics appropriately (for example, it may provide information about the Unit ID to which it is assigned).

6 XGPL Policy Excerpts

In this section we provide a collection of policy excerpts that illustrate some capabilities of the XG policy language. In the following subsections, we describe policy examples in English, and then provide annotated encodings of the examples in the XG policy language using the shorthand notation described in Section **Error! Reference source not found.**.

Note that these examples are notional (as with other examples in this document), and are intended only to illustrate the features of the language. Also note that the examples are brief excerpts and not complete or usable policies. They depend on, and relate to, other background facts that we have omitted for brevity. A complete annotated example is provided later in Appendix **Error! Reference source not found.**.

6.1 Geographic-Based

A policy rule can include geographic or spatial constraints in its selector, opportunity, or usage descriptions. For instance, we can set the `regionDesc` part of a policy rule's selector to specify that the policy rule applies to a particular country or region.
This example will define a selector description that applies to all locations within 30 miles of the geographic coordinates: 42° 21' 30"N, 71° 03' 37" W.

No.	Encoded Policy in Shorthand Notation	Remarks

1	`(SelDesc (id S1)`
	`(authDesc US-FCC)`
	`(freqDesc F1)`
	`(regnDesc US-MA-BostonDesc)`
	`(timeDesc Forever)`
	`(devcDesc D1))`
		A selector description that specifies a region description named BostonDesc (Line 2)

2	`(RegionDesc (id US-MA-BostonDesc)` `(region US-MA-Boston))`	A region description with one region named Boston (Line 3)

3	`(CylindricalArea (id US-MA-Boston)`
	`(centerAt BosCoord)`
	`(heightOf H3)`
	`(radiusOf R30))`
		A cylindrical area centered at BosCoord (Line 4) with a radius of 30 miles (Line 5) and a height of 3 miles (Line 6)

4	`(GeographicCoordinate`
	`(id BosCoord)`
	`(latitudeOf 42.358333)`
	`(longitudeOf −71.060278))`
		A geographic coordinate in decimal form

5	`(Radius (id R30)` `(magnitude 30) (unit mi))`	A parameter specifying the radius
6	`(Height (id H3)` `(magnitude 3) (unit mi))`	A parameter specifying the height

6.2 Time-Based

A policy rule may involve time in its selector, opportunity, or usage descriptions. This section will illustrate two examples of policies involving time.

First, a policy rule can be limited to a particular time period by setting the `timeDesc` part of a policy rule's selector. ***This example will define a selector description that applies to just the year 2004.***

No.	Encoded Policy in Shorthand Notation	Remarks

1	`(SelDesc (id S2)`
	`(authDesc US-FCC)`
	`(freqDesc F1)`
	`(regnDesc US)`
	`(timeDesc Year2004desc)`
	`(devcDesc D1))`
		A selector description that specifies a time description named Year2004desc (Line 2)

2	`(TimeDesc (id Year2004desc)` `(time Year2004))`	A time description that includes the time interval Year2004 (Line 3)

3	`(TimeInterval (id Year2004)`
	`(starttime start2004)`
	`(endtime end2004))`
		A time interval that starts at start2004 (Line 4) and ends at end2004 (Line 5)

4	`(DateTime (id start2004)`
	`(tdyear 2004) (tdmonth 1)`
	`(tdday 1) (tdhour 0)`
	`(tdminute 0) (tdsecond 0))`
		A time structure that specifies the start of the year 2004: midnight January 1

5	`(DateTime (id end2004)`
	`(tdyear 2004) (tdmonth 12)`
	`(tdday 31) (tdhour 23)`
	`(tdminute 59) (tdsecond 59))`
		A time structure that specifies the last second of the year 2004: December 31, 23:59:59.

Additionally, time-based constraints may be expressed as part of the opportunity or usage constraint description. ***This example will define policy rules that limit the maximum continuous transmission on time to 1 second and the minimum off time to 100 msec.***

No.	Encoded Policy in Shorthand Notation	Remarks

1	`(PolicyRule (id P1) (selDesc S1`
	`(deny FALSE) (oppDesc AnyOpp)`
	`(useDesc U1))`
		For operation matching selector S1 (defined elsewhere) allow use subject to constraints specified in U1 (Line 5).

2	`(PolicyRule (id P2) (selDesc S1`
	`(deny FALSE) (oppDesc AnyOpp)`
	`(useDesc U2))`
		For operation matching selector S1 allow use subject to constraints specified in U2 (Line 6).

3	`(TimeDuration (id OnTime1)` `(magnitude 1.0) (unit sec))`	**OnTime1 = 1 sec**
4	`(TimeDuration (id OffTime1)` `(magnitude 100.0) (unit msec))`	**OffTime1 = 100 msec**
5	`(UseDesc (id U) (xgx "(<=` `Emission.OnTime OnTime1)"))`	**Emission.OnTime <= OnTime1**
6	`(UseDesc (id U2) (xgx "(>=` `Emission.OffTime OffTime1)"))`	**Emission.OffTime >= OffTime1**

6.3 Device Capability-Based

The XG policy language can express policies that apply to particular types of devices, or to devices with certain capabilities. For example, policies could be written so that devices with particular capabilities would have more opportunities to use the spectrum than those that do not have those capabilities. This can be expressed by specifying the device in the selector description, as well as in the opportunity and usage constraint descriptions.

This example will define a selector description that applies to a radio that is certified as being XG version 1 compliant, which means that the device supports a certain set of parameter and process types. The parameters and processes must be described within the ontology, but their descriptions are not shown in the example.

No.	Encoded Policy in Shorthand Notation	Remarks
1	(SelDesc (id S1) (authDesc US-FCC) (freqDesc F1) (regnDesc US) (timeDesc Forever) (devcDesc D1))	A selector description that specifies a device description named D1 (Line 2)
2	(DeviceDesc (id D1) (deviceTyp XGv1) (deviceCap Profile1))	A device description that specifies a device with type XGv1 and capabilities defined in Profile1
3	(DeviceTyp (id XGv1))	A device type named XGv1. The meaning of the name must be defined by the authority and grounded by the device.
4	(DeviceCap (id Profile1) (hasPolicyDefinedParams EmissionFrequencyRange CurrentLocation MaxTransmitPower PowerLeakage) (hasPolicyDefinedBehaviors Listen Sub Distance))	This defines the capabilities of a device. This device must understand and be able to bind the parameters: EmissionFrequencyRange, CurrentLocation, MaxTransmitPower, and PowerLeakage and understand and be able to ground the processes: Listen, Sub, and Distance

6.4 Explicit Grouping of Policy Rules

Policy rules can be grouped together by explicitly stating the policy rules that are part of the group. Other policy statements can then refer to this group as a whole. ***This example shows how the following three policy rules are grouped together***:

- ***Transmission is limited to the band 3100 MHz to 3300 MHz.***
- ***If the peak received power in sensing is less than –80 dBm, the maximum EIRP for transmission is 10 mW.***
- ***If the peak received power is greater than –80 dBm, the device must not transmit.***
-

No.	Encoded Policy in Shorthand Notation	Remarks
1	`(PolicyGrp (id G1) (polMembers P1 P2 P3))`	A policy group consisting of policy rules P1, P2, and P3
2	`(PolicyRule (id P1) (selDesc S1) (deny FALSE) (oppDesc AnyOpp) (useDesc U1))`	For operation matching selector S1 (defined elsewhere), allow use subject to constraints specified in U1 (defined elsewhere), namely transmission in the 3100-3300MHz band.
3	`(PolicyRule (id P2) (selDesc S1) (deny FALSE) (oppDesc O2) (useDesc U2))`	For operation matching selector S1, allow use of opportunity defined if O2 is satisfied (Line 7), subject to usage constraints U2 (Line 8).
4	`(PolicyRule (id P3) (selDesc S1) (deny TRUE) (oppDesc O3) (useDesc DenyUse))`	For operation matching selector S1, deny use of spectrum if O3 is satisfied (i.e. Peak power is greater than –80dBm). This rule is not needed if an overarching default policy that denies use has been specified.
5	`(Power (id Sense1) (magnitude –80.0) (unit dBm))`	A parameter specifying sensed power
6	`(Power (id Transmit1) (magnitude 10.0) (unit mW))`	A parameter specifying transmit power
7	`(OppDesc (id O2) (xgx "(< PeakSensedPower Sense1)"))`	PeakSensedPower < Sense1
8	`(UseDesc (id U2) (xgx "(<= MaxTransmitPower Transmit1)"))`	MaxTransmitPower <= Transmit1

6.5 Grouping of Policy Rules using Expressions

Groups can also be formed using an expression that defines membership in the group. ***In this example, we create a group that consists of all policies with a selector description of S3 (where S3 is described the same way as selector S1 in the previous example, but requires the additional capability of detecting the XYZZY waveform). Then we create the following policy rule uses S3:***

- ***If peak received power is greater than –80dBm and less than –50dBm and the XYZZY waveform is not detected, then the maximum transmit EIRP is 10mW.***

No.	Encoded Policy in Shorthand Notation	Remarks
1	`(PolicyGrp (id G2) (xgx "(selMatches S3)"))`	A policy group consisting of all policy rules with a selector description that matches S3
2	`(PolicyRule (id P4) (selDesc S3) (deny FALSE) (oppDesc O4) (useDesc U4))`	For operation matching selector S3 (defined elsewhere) allow use of opportunity defined if O4 is satisfied (Line 6), subject to usage constraints U4 (Line 7).
3	`(Power (id Sense1) (magnitude -80.0) (unit dBm))`	A parameter specifying sensed power
4	`(Power (id Sense2) (magnitude -50.0) (unit dBm))`	A parameter specifying sensed power
5	`(Power (id Transmit1) (magnitude 10.0) (unit mW))`	A parameter specifying transmit power
6	`(OppDesc (id O4) (xgx "(and (> PeakSensedPower Sense1) (< PeakSensedPower Sense2) (invoke DetectWaveform1 Waveform XYZZY Presence WFPresent) (eq WFPresent BoolFalse))"))`	`PeakSensedPower > Sense1, and PeakSensedPower < Sense2, and` The opportunity condition of accounting approval is determined by invoking the DetectWaveform1 process (that is specified elsewhere, and must be grounded by the radio) with the waveform set to XYZZY as an input parameter and its presence or absence as an output parameter.
7	`(UseDesc (id U4) (xgx "(<= MaxTransmitPower Transmit1)"))`	MaxTransmitPower <= Transmit1

6.6 Precedence and Default Rules

Precedence allows the policy administrator to specify how to resolve policy conflicts. It is also useful to specify restrictive default rules that can be relaxed later by specifying other rules that enable specific opportunities and constraints on the use of those opportunities. ***In this example, we specify a default rule P_default (placed within a group G_default) that denies all use if selector S2 is matched. Then, we specify a meta-policy that provides the policy group G2 (from the previous example) a higher precedence over the default rule, thereby enabling operation under the policy rules in G2.***

No.	Encoded Policy in Shorthand Notation	Remarks
1	`(PolicyRule (id P_default) (selDesc S2) (oppDesc AnyOpp) (useDesc DenyUse) (deny TRUE))`	**This defines a policy that denies emissions under all circumstances for selDesc S2. `AnyOpp` (description not shown) always evaluates to true.**
2	`(PolicyGrp (id G_default) (polMembers P_default))`	**Create a group that contains the policy rule P_default.**
3	`(PolGrpPrecedes (left G2) (right G_default))`	**The group G2 precedes the group G_default. This indicates that the policy rules that are members of G2 have precedence over those in G_default, and therefore operation will be permitted subject to the rules in G2, overriding the default policy that does not allow any transmission.**

6.7 Disjunction

Disjunction of a list of policy groups can be used to specify that any one (or more) of the enumerated groups of policies can be applied at a time. ***In this example, we specify the disjunction of G1 and G2 (from earlier examples) and allow the radio to choose which group it wishes to operate under.***

No.	Encoded Policy in Shorthand Notation	Remarks
1	`(DisjunctGrps (polGroups G1 G2))`	A meta-policy that states that the policy rules in group G1 are disjunct from the policy rules in group G2, so a radio needs to use the policy rules from only one of the groups.

6.8 Policies for Secondary Markets/Pass-through

XGPL can express policies that enable a hierarchy of policy authorities and primaries to lease part of their allocated spectrum to other users. This includes support for a policy authority to delegate part of the spectrum that it has jurisdiction over to a sub-authority or a primary user. While the sub-authority is free to specify its own policies for the spectrum it has been delegated, clearly its policies may not violate any policies of the super-authority unless explicitly permitted.

XGPL enables specification of policies for both the super- and sub-authorities. The super-authority may use grouping and precedence rules to ensure that its policy rules have precedence over the sub-authority's rules. With these meta-policy rules in place, the normal policy processing rules will ensure that rules from both authorities will be enforced if there are no conflicts; in the case of conflict, the super-authority's rules prevail.

In this example, there are two authorities, A1 and A2. A2 is leasing some of the frequencies it's been allocated to XG devices. A1 and A2 have the following policies:

- ***A1 specifies the maximum transmit power in A2's spectrum is 10mW***
- ***A2 specifies that within its allocated spectrum XG devices must get approval from A2's accounting servers before they may use the spectrum.***
- ***A2 specifies that within its allocated spectrum the maximum transmit power is 15mW***

The end result of these policies is that an XG device may transmit up to 10mW in A2's spectrum, if it has permission to do so from A2's accounting server.

We will start with the policy specified by authority A1:

No.	Encoded Policy in Shorthand Notation	Remarks

1	`(SelDesc (id S4)`
	`(authDesc A1)`
	`(freqDesc F1)`
	`(regnDesc Boston)`
	`(timeDesc Forever)`
	`(devcDesc Any))`
		A selector description that specifies A1 as the authority.

2	`(PolicyGrp (id PG1)` `(polMembers P6))`	A policy group consisting of policy rule P6
3	`(PolicyGrp (id PG2)` `(xgx "(authMatches A2)"))`	A policy group consisting of all policy rules with A2 as an authority.
4	`(PolGrpPrecedes` `(left PG1) (right PG2))`	A meta-policy that specifies the policy rules in group PG1 have precedence over the policy rules in group PG2.

5	`(PolicyRule (id P6) (selDesc S4)`
	`(deny FALSE) (oppDesc AnyOpp)`
	`(useDesc U6))`
		For operation matching selector S4 (Line 1), allow use subject to usage constraints U6 (Line 7).

6	`(Power (id Transmit1)` `(magnitude 10.0) (unit mW))`	A parameter specifying transmit power.
7	`(UseDesc (id U6) (xgx "(<=` `MaxTransmitPower Transmit1)"))`	MaxTransmitPower <= Transmit1

Next, we present the encoding of the sub-policy specified by authority A2:

No.	Encoded Policy in Shorthand Notation	Remarks
1	(SelDesc (id S5) (authDesc A2) (freqDesc F1) (regnDesc Boston) (timeDesc Forever) (devcDesc XG))	A selector description that specifies A2 as the authority.
2	(PolicyRule (id P7) (selDesc S5) (deny FALSE) (oppDesc O7) (useDesc U7))	For operation matching selector S5 (Line 1), allow use of frequency band F1 if O7 is satisfied (i.e. Accounting approval), subject to usage constraints U7 (Line 5)
3	(Power (id Transmit2) (magnitude 15.0) (unit mW))	A parameter specifying transmit power.
4	(OppDesc (id O7) (xgx "(and (invoke CheckAccount Location CurrLocation Approval Approved) (eq Approved BoolTrue))"))	The opportunity condition of accounting approval is determined by invoking the CheckAccount process (that is specified elsewhere, and must be grounded by the radio) with the current location as an input parameter and the approval as an output parameter.
5	(UseDesc (id U7) (xgx "(<= MaxTransmitPower Transmit2)"))	MaxTransmitPower <= Transmit2

6.9 Public Safety/Interruptible[18] Spectrum

If XG devices were to be allowed to use public safety spectrum, then it is conceivable that regulatory policies would have to be in place that require the XG device to vacate the spectrum on a moment's notice. One possible implementation could use beacons–a *permit-use* ("take it") beacon and possibly a *release* ("leave it") beacon–that signal whether or not the spectrum is available.

In this example, we have a permit-use beacon at 823MHz. If it is heard, the radio may transmit; if not, the radio may not transmit. The parts of the policy that might specify constraints on the radio's emissions are not shown in this excerpt, as this example is focused on how an opportunity can be described in this context.

No.	Encoded Policy in Shorthand Notation	Remarks
1	(PolicyRule (id P10) (selDesc S1) (deny FALSE) (oppDesc O10) (useDesc U10))	For operation matching selector S1 (defined elsewhere), allow use of public safety spectrum if O10 is satisfied (i.e. beacon is heard), subject to usage constraints U10 (not shown here)
2	(PolicyRule (id P11) (selDesc S1) (deny TRUE) (oppDesc O11) (useDesc DenyUse))	For operation matching selector S1, deny use of spectrum if O11 is satisfied (i.e. when beacon is not heard). This rule is not needed if an overarching default policy that denies use has been specified.
3	(Boolean (id BeaconPresent1))	Parameter to indicate if beacon is present
4	(Frequency (id BeaconFreq1) (magnitude 823.0) (unit MHz))	A parameter specifying the beacon frequency
5	(SignalEncoding (id BeaconSig1))	A parameter specifying the beacon signal encoding.
6	(OppDesc (id O10) (xgx "(and (invoke SenseSignal BeaconFreq BeaconFreq1 BeaconEnc BeaconSig1 BeaconPresent BeaconPresent1)) (eq BeaconPresent1 BoolTrue))"))	The opportunity condition of beacon is present as determined by invoking the SenseSignal process (that is specified elsewhere, and must be grounded by the radio) with the beacon frequency and beacon signal encoding as input parameters, and the Boolean variable BeaconPresent1 as output.
7	(OppDesc (id O11) (xgx "(and (invoke SenseSignal BeaconFreq BeaconFreq1 BeaconEnc BeaconSig1 BeaconPresent BeaconPresent1) (eq BeaconPresent1 BoolFalse))"))	An opportunity condition of beacon not present as determined by invoking the SenseSignal process as in the previous line.

[18] The need for interruptible operation can arise in other scenarios as well.

6.10 Re-Use of Television Bands

Suppose a regulatory authority decides to provide spectrum-sharing rights to XG users on a non-interference basis within bands originally allocated for television broadcasts. Initially, policies may be provided within limited geographical regions, and with temporal restrictions. Furthermore, these policies may be limited to a small number of channels, for example, selected UHF TV channels. As experience is gained with spectrum sharing policies and technologies through extensive experimentation and field-testing, the policies and the technologies will both evolve. Policies can be crafted to expand the opportunities available to sophisticated receivers capable of sub-noise detection of a DTV signal. A notional policy example governing this scenario is provided below. Note that in this example, an XG radio with the ability to perform sub-noise detection of DTV signals is given rights to transmit at a higher power spectral density as long as the DTV signal is detected below the noise threshold.

In this example, the following policies apply to all TV bands:

- ***Incumbent primaries in TV bands may broadcast NTSC, PAL, or DTV signals.***
- ***The XG radio may transmit on at most 6 MHz at a time.***

In this example, the following policies provide transmission rights for XG radios that are capable of operating in TV bands 60-69:

- ***This policy expires at 00:00 hrs GMT on April 1, 2005.***
- ***The use of XG radios in these bands is limited areas governed by USA spectrum policies.***
- ***The XG radio sensor must sample at least once every 5 seconds.***
- ***The continuous on time of the XG radio must not exceed 1 second, the off time must not be less than 150 milliseconds, and duty cycle must not exceed 50% across a 2 second period.***
- ***The emission power of the XG radio must exceed -60dBm/Hz outside of the intended channel***
- ***If the XG radio uses power sensing (no sub-noise detection of DTV signals):***
    - ***The XG Radio may transmit on channels where the received spectral power in the channel is less than -100dBm.***
    - ***The peak power spectral density of the XG radio's emission shall not exceed -53 dBm/Hz.***
- ***If the XG radio uses the capability of sub-noise detection of DTV signals:***
    - ***The XG radio may transmit on a channel in which the received DTV signal power is less than -107 dBm***
    - ***The peak power spectral density of the XG radio's emission shall not exceed -53 dBm/Hz plus 1 dB for every 1 dB the received signal is below –107dBm up to a maximum of -40 dBm/Hz.***

No.	Encoded Policy in Shorthand Notation	Remarks
1	(SelDesc (id S9) (authDesc US-FCC) (freqDesc TVBands) (regnDesc US) (timeDesc Forever) (devcDesc XG))	A selector description that specifies all television bands and all XG devices for an unlimited amount of time. For this example only, we assume all XG devices
2	(SelDesc (id S10) (authDesc US-FCC) (freqDesc TV60-69) (regnDesc US) (timeDesc Until-040105) (devcDesc XG))	A selector description that specifies only TV channels 60-69 and a time period from the present to April 1, 2005 for all XG devices.
3	(SelDesc (id S11) (authDesc US-FCC) (freqDesc TV60-69) (regnDesc US) (timeDesc Until-040105) (devcDesc XG-subnoise))	A selector description that specifies only TV channels 60-69 and a time period from the present to April 1, 2005 for all XG devices that can have the capability of sub-noise detection for DTV signals.
4	(PolicyRule (id P1) (selDesc S10) (oppDesc AnyOpp) (deny FALSE)(useDesc U_TV60-69))	A policy rule that states that for selector S10 (Line 2), transmission is limited to TV channels 60-69 (U_TV60-69 defined elsewhere).
5	(PolicyRule (id P2) (selDesc S9) (oppDesc AnyOpp) (deny FALSE) (useDesc U_BW6MHz))	A policy rule that states that for selector S9 (Line 1), transmission in any TV channel is limited to a 6 MHz bandwidth (U_BW6MHz defined elsewhere).
6	(PolicyRule (id P3) (selDesc S10) (oppDesc O_Sense5sec) (deny TRUE) (useDesc DenyUse))	A policy rule that states that for selector S10 (Line 2), there is no opportunity if sensing is done less than one every 5 seconds (O_Sense5sec defined elsewhere).
7	(PolicyRule (id P4) (selDesc S10) (oppDesc AnyOpp) (deny FALSE) (useDesc U_OnTime1))	A policy rule that states that for selector S10 (Line 2), on time must not exceed 1 second(U_OnTime1 defined elsewhere).
8	(PolicyRule (id P5)(selDesc S10) (oppDesc AnyOpp) (deny FALSE) (useDesc U_OffTime1))	A policy rule that states that for selector S10 (Line 2), off time must not be less than 150 msecs (U_OffTime1 defined elsewhere).
9	(PolicyRule (id P6) (selDesc S10) (oppDesc AnyOpp) (deny FALSE) (useDesc U_DutyCycle50Pc2s))	A policy rule that states that for selector S10 (Line 2), the duty cycle must not exceed 50% across a 2 s time period (U_DutyCycle50Pc2s defined elsewhere).
10	(PolicyRule (id P7) (selDesc S10) (oppDesc AnyOpp) (deny FALSE) (useDesc U_Leakage-60dBmperHz))	A policy rule that states that for selector S10 (Line 2), emissions outside the intended channel must not exceed -60 dBm/Hz (U_Leakage-60dBmperHz defined elsewhere).
11	(PolicyRule (id P8) (selDesc S10) (oppDesc O_Power-100dBm) (deny FALSE) (useDesc U_PSD-53dBmperHz))	A policy rule that states that for selector S10 (Line 2), there is an opportunity if received power is less than -100dBM (O_Power-100dBm defined elsewhere) and the emission is less than -53 dBm/Hz (U_PSD-53BmperHz defined elsewhere).
12	(PolicyRule (id P9) (selDesc S11) (oppDesc O_Power-107dBm) (deny FALSE) (useDesc U_PSD-53dBmperHzplus))	A policy rule that states that for XG radios have the capability of sub-noise detection (selector S11, Line 2), there is an opportunity if received power is less than -107dBM (O_Power-107dBm defined elsewhere) and the emission is limited as specified in the policy rule (Line 13).

13	(UseDesc (id U_PSD-53dBmperHzplus) (xgx "(and (<= Emission.PSD PSD-40dBmperMhz) (<= Emission.PSD (+ PSD-53dBmperHz (- Pwr-107dBm SensedSignal.Power))))) "))	A usage constraint stating: • Emission.PSD <= -40 dBm/Hz • Emission.PSD <= -53 dBm/Hz + -107dBm - SensedSignal.Power
14	(PolicyGrp (id G1) (polMembers P8))	A policy group containing policy rule P8.
15	(PolicyGrp (id G1) (polMembers P9))	A policy group containing policy rule P9.
16	(DisjunctGrps (polGroups G1 G2))	A meta-policy that states that the policy rules in group G1 are disjunct from the policy rules in group G2, so an XG radio needs to either use the rule using sub-noise detection or use the rule without it.

7 Discussion

In this section we discuss:

- **Related work**: a brief discussion of related work that has influenced this document
- **Future Work:** plans for refining and extending the concepts described in this document (to be incorporated in future versions)
- **Summary:** a summary of the major contributions of this document

7.1 Related Work

The policy language presented in this document was influenced by other work in the policy community and previous policy work done at BBN. We describe that work in this section. We note, however, that beyond the work discussed here, there is an extensive body of literature on languages for expressing policy, algorithms to interpret policy and check conformance, algorithms to determine optimal solutions for operation under policy constraints, and systems to manage, monitor, provision, and enforce policy in various domains.

Mitola and Maguire [MITOL] investigated languages to represent policy for cognitive radios. They conclude that existing languages are not sufficient to express the knowledge required and developed the Radio Knowledge Representation Language (RKRL). However this work was performed before OWL was developed to provide a standards-based solution to the problem.

OWL's precursor DAML has been used previously to represent policy to support logistics in KaoS Domain and Policy Services [KAOS]. It provides a source on the use of DAML and may be the source of some DAML tools. KaoS also contains an authorization/obligation model that may be useful for XG.

As part of the DARPA-funded Policy Based Security Management [PBSM] project, BBN developed a policy language and a policy discovery and exchange protocol for IPsec communications. BBN's work in this area led to the formation of the IPsec Policy [IPSP] working group in the IETF that influenced the Policy Framework [PCIM] working group. PCIM developed a general framework for policy that has been extended into various policy domains including security and quality of service. BBN also developed an XML-based policy language for communicating mission-related security requirements among the dynamic membership of a coalition in the DARPA-funded Multi-Dimensional Security Policy Management and Enforcement [MSME] project.

There are tradeoffs between the expressiveness and the processing complexity of a policy specification language. Griffin *et al.* [GJF03] address this issue by laying out the design principles of policy languages, and include rigorous analyses in the context of path vector protocols such as BGP.

Finally, the KeyNote/PolicyMaker system [KeyNote] describes an authority-delegation model that may be useful for delegation of authorization in XG and it provides insights on the issues for providing polynomial time conformance verifiability.

7.2 Future Work

This is an initial version of the XG policy language framework. We plan to refine and extend this work within the XG program, based in part on the feedback we receive from the community on this RFC. In particular, the extensions may include:

- Extensions to the XGPL ontologies to include a wider range of concepts used to express spectrum policy as well as background facts such as frequency band classifications
- An ontology that describes the capabilities of a notional XG radio and that can be used to further illustrate policy agile radio behaviors within an XG environment
- Basic tools and utilities for processing policy, including an interpreter, format converter, and a policy conformance reasoner
- Additional examples that illustrate the use of the XG policy language
- Tradeoff analyses of processing complexity versus expressive power for the XG policy language, as a guidance to policy writers on how to structure policy
- Development of the policy processing logic, including rules for processing meta-policies, handling of delegation of authority, and decorrelation of selector instances
- Visualization and editing tools
- Modeling better paradigms and idioms that fit better with concerns of policy-makers, for example, rights-limits or authorization-obligation patterns
- Engage the regulatory community, radio vendors, wireless service providers, and standards organizations for further development and adoption of this promising technology

Future revisions of this document may include the results of such work, as appropriate.

7.3 Summary

Radio devices are becoming more agile and programmable. The regulatory community is moving toward opening up opportunistic access to spectrum. These trends are poised to create a new era in wireless communications in which spectrum scarcity is no longer an issue and zero deployment time is a reality. This will have a great impact in both the commercial and military sectors. The XG program is pioneering the field of opportunistic spectrum access by developing key technologies that enable efficient and practical use of spectrum sharing across regulatory policy regimes.

A key technological challenge in this environment is how to exploit the emerging agility of radio devices while retaining the ability to accredit them for policy conformance under all possible modes of operation, different operating environments, and variable regulatory jurisdictions. Hard coding policy sets into radios for all possible combinations will become impractical, as will the maintenance and management of accredited software

implementations for all the distinct combinations of policy sets and target platforms. Furthermore, as technology often gets developed in advance of policy, hard-coding policy into the radios can result in costly reengineering when the policies change.

The key to addressing this problem is a framework that permits the control of radio behaviors in-situ using a machine-understandable policy language. In other words, radios must be *policy-agile* in addition to being hardware agile (e.g. frequency-agile, or waveform-agile).

In this RFC, we propose a framework to express and process policy in order to enable radios to be policy-agile. We describe a concept of operations for how machine-understandable policy can be created, as well as how a radio can make use of this encoded policy in order to be situationally adaptive to policy. Our framework enables the definition of a well-defined *accreditation boundary*, which is similar to the concept of a "trusted kernel" in secure systems. Critical policy conformance validation functions (that are within regulatory purview) are kept within this boundary. At the same time, all system-dependent innovations, optimizations, and engineering tradeoffs are outside this boundary, providing a clean separation of concerns.

We present a language to express policy, including the terms, their meanings, their interrelationships, and useful constructs and idioms. In our approach, the policy does not tell the radio what to do; rather, it specifies what constitutes conforming use of spectrum. A key feature of our approach is the use of a rule-based rather than a procedural approach. This approach enables the specification of policies in a manner that is both more scalable and more maintainable over time—because new parameters and policies can be added without modifying previously specified policies.

Moreover, by using an extensible knowledge-representation framework, we have allowed for the language to evolve to meet the needs of policy concepts and technological concepts that are yet to be developed.

The use of a standards-based representation, namely, the W3C's OWL, will enable wider adoption, make a larger base of tools available, and leverage the benefits of decades of research and development efforts in the areas of markup languages for the World Wide Web and knowledge representation.

As proof-of-concept, we present a number of examples that illustrate the capabilities of the language to express *notional* policies, including policies that are time-dependent and location-dependent, and policies that may address proposed spectrum-sharing concepts such as dynamic spectrum allocation based on sensing incumbent signals, secondary spectrum markets with sub-policy management, and the interruptible use of public safety spectrum. In the appendices, we present a complete example, and provide links to related OWL ontologies that are available online.

We believe this is a promising approach that can enable the practical use of spectrum sharing technologies across regulatory policy regimes. We are in the early stages of the

evolution of this technology. We hope to engage the regulatory community, radio vendors, wireless service providers, and standards organizations for further development and adoption of this promising technology.

To conclude, this document is a Request for Comments. We actively seek input and feedback from the community.

Acknowledgments

This document was prepared by the Internetwork Research Department, BBN Technologies, with input from DARPA/ATO, AFRL, Alion Science, the XG System Integration teams, and Mike Dean.

Comments

Comments on this RFC, along with the commenter's name and organization, should be emailed to the document editors at xg-rfc-comments@bbn.com

Bibliography

[ARGO] ArgoUML http://argouml.tigris.org/
[BKK01] Kenneth Baclawski, *et al*., "Extending UML to Support Ontology Engineering for the Semantic Web," Proceedings of the Fourth International Conference on UML (UML 2001), Toronto, Oct 1-5, 2001
[CLIPS] C Language Integrated Production System http://www.ghg.net/clips/CLIPS.html
[COMP] http://www.daml.org/language/features.html
[DAML] DARPA Agent Markup Language, http://www.daml.org/
[DAMLOT] A DAML Ontology of Time, http://www.cs.rochester.edu/~ferguson/daml/
[GJR03] Timothy G. Griffin *et al*, "Design Principles of Policy Languages for Path Vector Protocols," ACM SIGCOMM, Karlsruhe, Germany, Aug 25-29, 2003. pp.61-72.
[IPSP] IETF Security Policy Working Group, http://www.ietf.org/html.charters/ipsp-charter.html
[ISO8601] International Organization for Standardization, "Data elements and interchange formats–Information interchange–Representation of dates and times," ISO 8601:2000.
[KAOS] The KaoS Ontology and Policy Management System, http://ontology.coginst.uwf.edu/kaos.html
[KeyNote] The KeyNote Trust Management System, http://www.cis.upenn.edu/~keynote/
[MITOL] Joseph Mitola III and G. Q. Maguire Jr., "Cognitive Radio: Making Software Radios More Personal," IEEE Personal Communications, Aug 1999, pp. 13-18.
[MSME] Multi-dimensional Security Management and Enforcement, http://www.ir.bbn.com/projects/msme/
[OWL] OWL Web Ontology Language, http://www.w3.org/2001/sw/WebOnt/
[PBSM] Policy-based Security Management, http://www.ir.bbn.com/projects/pbsm/pbsm-index.html
[PCIM] IETF Policy Framework Working Group, http://www.ietf.org/html.charters/policy-charter.html
[RDF] Resource Description Framework, http://www.w3.org/RDF/
[RLXNG] RELAX NG schema language for XML http://relaxng.org/
[SOX] Schema for Object-Oriented XML 2.0, W3C Note 30 Jul 1999, http://www.w3.org/TR/NOTE-SOX
[SWRL] Semantic Web Rule Language, http://www.daml.org/rules/proposal/
[ULTLG] The DARPA UltraLog Program. http://www.ultralog.net/
[UML] Unified Modeling Language, http://www.uml.org/
[XGV] The XG Vision RFC, http://www.darpa.mil/ato/programs/XG/rfc_vision.pdf
[XGAB] The XG Abstract Behaviors RFC (under preparation).
[XGAF] The XG Architectural Framework RFC, http://www.darpa.mil/ato/programs/XG/rfc_af.pdf
[XSD] XML Schema Part 2: Datatypes, http://www.w3.org/TR/xmlschema-2/

A Terminology and Acronyms

A list of definitions of terminology and acronyms used in this document is provided here for convenience.

A.1 Glossary

Binding: Associating a value with a **parameter**. The value of a parameter may not be known when the policy is written, for example, values that must be provided by an XG radio when the policy is processed.

Fact: A statement of policy knowledge. Policy consists of a set of facts and associated rules for processing.

Grounding: An implementation of **process** keywords in a policy. Many keywords in the policy language refer to a function that must be implemented by the radio. The XG radio platform must implement (i.e., provide a grounding for) these keywords to be able to use policies that refer to them.

Instance: A set of bound parameters that can be used to determine the truth-value of a selector, opportunity or usage description.

Machine Understandable Policies: Policies expressed in a form that allows for an XG radio to automatically (without requiring human intervention) read and "interpret" them. That is, there exists an automated procedure by which the implications of the constraints expressed by the policies are reflected in the actions of the radio.

Meta-policy: A **fact** that states relationships between **Policy-Rule**s.

Ontology: The representation of terms in a vocabulary and their inter-relationships.

Opportunity Description: The second fact in a **Policy Rule**; it provides an expression that is used to evaluate whether or not a given environment and device state represents an opportunity for the device to take action in accordance with the policy rule.

Parameter: A **fact** that describes a policy concept that can be associated with a value. Parameters may be bound or unbound.

Policy: a set of **Fact**s and associate **Rule**s that specify how a resource (spectrum in the case of this document) may be used.

Policy Rule: A statement of policy consisting of a set of facts, but not the rules for interpreting. A policy rule has a **Selector Description**, an **Opportunity Description** and a **Usage Description**.

Process: A **fact** that describes a function implemented by a radio. The description includes inputs and output parameters (analogous to a function prototype in languages such as C) and expressions constraining the relationship between the inputs and outputs.

Processes do not refer to operating system processes. A process may, for example, implement a *k*-ary relationship between parameters. An XG device must ground relevant process keywords with a function implementation that matches the description.

Regulatory Policy: A policy that is specified, or likely to be specified by a regulatory authority (such as the FCC or the NTIA in the U.S.A.). Typically, these describe what constitutes valid use of spectrum rather than provide specific instructions to the device on what specific actions to take.

Rule: A statement that describes the logic for interpreting and processing policy. Rules have the form: *condition-implies-action.*

Selector Description: The first **fact** in a **Policy Rule**; it is used to filter policies to obtain the set that may apply to an intended operating environment or situation (described by frequency, time, region, and device characteristics).

Spectrum Overlay: Using adaptive spectrum access techniques to identify underutilized spectrum and to avoid interference conflicts in time, frequency or space with competing spectrum users. Unlicensed spectrum users have used these techniques (e.g., 802.11a Dynamic Frequency Selection) to share spectrum with incumbent licensed users.

Spectrum underlay: Simultaneous use of spectrum in time and frequency by multiple uncoordinated emitters that takes advantage of modulation techniques such as spread spectrum or ultra-wideband to limit interference between systems. Transmitter power output may be restricted to further limit the possibility of interference. Typically at least one of the emitters is a spread spectrum signal with a large amount of processing gain to insure that the undesired signal power seen by an incumbent licensed user is below a designated threshold.

System Policy: A policy representing dynamic, location specific, or capability based guidance, intended to constrain and influence XG radio behaviors, decisions, and actions. The system policy is likely specified by a system administrator and typically specifies inputs beyond those available in regulatory policy. It can provide specific strategies or instructions to the radio.

Usage Description: The third **fact** in a **Policy Rule**; it provides an expression that constrains XG spectrum use behavior such as the emission permitted and the corresponding sensing requirements when using the opportunity described in the policy rule.

A.2 Acronyms

BGP: Border Gateway Protocol
BNF: Backus-Naur Form
CLIPS: C Language Integrated Production System
DTD: Document Type Declaration
DAML: DARPA Agent Markup Language

DARPA: Defense Advance Research Projects Agency
DSP: Digital Signal Processor
DTV: Digital Television (television standard)
EIRP: Effective Isotropic Radiated Power
FCC: Federal Communications Commission
HTML: Hyper-Text Markup Language
IETF: Internet Engineering Task Force
KQML: Knowledge Query and Manipulation Language
KR: Knowledge Representation
NTIA: National Telecommunications and Information Administration
NTSC: National Television System Committee (television standard)
OIL: Ontology Inference Layer
OWL: Web Ontology Language
PAL: Phase Alternating Line (television standard)
RDF: Resource Description Framework
RF: Radio Frequency
RFC: Request for Comments
RKRL: Radio Knowledge Representation Language
SI: International System of Units
SOX: Schema for Object-Oriented XML
UML: Unified Modeling Language
URI: Uniform Resource Identifier
URL: Uniform Resource Locator
W3C: World Wide Web Consortium
XG: DARPA's NeXt Generation Communications Program
XGPL: XG policy language
XG-WG: XG working group
XML: Extensible Markup Language

B URLs for XGPL Ontologies, Examples, and Software

B.1 XGPL Ontologies

This section provides the URLs of the OWL ontologies that implement the XG policy language as defined in this document. The sections refer to the section that defines the ontology in this document.

Section	*URL*
3.1	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-rl.owl
3.2	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl.owl
3.3	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-param.owl
3.4	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-proc.owl
3.5	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-auth.owl
3.6	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-freq.owl
3.7	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-regn.owl
3.8	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-time.owl
3.9	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-devc.owl
3.10	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-env.owl
3.11	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-sysext.owl
3.12	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-physq.owl
3.13	http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-unit.owl

B.2 Examples

The notional policy examples described in this document (in Section 6 and Appendix C) are available online, encoded in XG policy language, both using the notational shorthand used in this document, as well as in OWL. They can be accessed at the following URL: http://www.ir.bbn.com/projects/xmac/owl/2004/03/rfc-examples

B.3 Processing Rules

We plan to make available the policy processing rules described in Section 4 also encoded using in the XG policy language. When completed, these will be available at the following URL: http://www.ir.bbn.com/projects/xmac/owl/2004/03/processing-rules

B.4 Tools

We plan to make available utilities to process the XG policy language, for example, programs to convert between the shorthand notation used in this document and the OWL representation of the XG policy language. When completed, these will be available at the following URL: http://www.ir.bbn.com/projects/xmac/owl/2004/03/tools

C Example

In this appendix, we present the analysis and encoding of a notional policy. We analyze the example described in English and map it to our ontological framework. We then describe the encoding in our surface language and provide an instance that would satisfy the policy.

The full encoding of this example in the shorthand notation is available online at:
http://www.ir.bbn.com/projects/xmac/owl/2004/03/rfc-examples/example-appA.xgsl

The equivalent encoding of the same example in OWL is available online at:
http://www.ir.bbn.com/projects/xmac/owl/2004/03/rfc-examples/example-appA.owl

C.1 English Description

This example applies to XG devices that are capable of operating in the 3.6-3.7 GHz band. The NTIA is the authority over this band in the US. Only emissions permitted by the following rules are allowed in this band:

1. Transmissions shall be contained within 3.6 GHz to 3.7 GHz.
2. The sensor must have a minimum look-through interval of 3 seconds.
3. The peak power spectral density of the XG device shall not be more than 1 nW/Hz.
4. The XG device may transmit only on channels where an incumbent signal "INCSIG" has been detected and the peak sensed power on each channel is less than –100 dBm.
5. The device may transmit on at most 10 channels at the same time.
6. No more than 1% of the power must be outside of the bandwidth of the intended frequency channel
7. The continuous on time must not exceed 1 second and the off time must not be less than 100 msec.

C.2 Ontological Analysis

Here we present a line-by-line ontological analysis of the example. The first column contains English statements from above. The second column provides an ontological classification of the statement and any clarifying remarks. The final column cross-references the facts in the encoding provided below.

English	***Ontological Analysis***	***X-ref***
This example applies to XG devices that are capable of operating in the 3.6-3.7 GHz band	SelDesc:DevcDesc:XG SelDesc:FreqDesc:FrequencyRange Some of this information may be background knowledge.	F13, F33, F36
The NTIA is the authority over this band in the US.	SelDesc:AuthDesc:US-NTIA SelDesc:RegnDesc:US This information may be background knowledge.	F13, BGF
Only emissions permitted by the following rules are allowed in this band:	A default policy exists that denies use of this band if the rest of the policy rules do not apply.	F10, F11, F12
Transmissions shall be contained within 3.6 GHz to 3.7 GHz.	UseDesc:Emission.Band	F1, F16

The sensor must have a minimum look-through interval of 3 seconds.	OppDesc:Sensor.LookThrough	F6, F15
The peak power spectral density shall not be more than 1 nW/Hz.	UseDesc:Emission.PeakPSD	F2, F19
The XG device may transmit only on channels where the incumbent signal "INCSIG" has been detected and the peak sensed power on each channel is less than –100 dBm.	UseDesc:SensedSignal.Type UseDesc:SensedSignal.Power UseDesc:SensedSignal.ChannelNum The policy also requires the notion of associating a channel with its corresponding sensed signal type and power. This must be represented as a process for which the XG device must have a grounding to be able to use this policy. This requirement is reflected in the device description.	F8, F14, F17
The device may transmit on at most 10 channels at the same time.	UseDesc:Emission.Channels	F7, F18
No more than 1% of the power must be outside of the bandwidth of the intended frequency channel	UseDesc:Emission.PowerLeakage	F5, F22
The maximum continuous on time must be 1 second and the minimum off time must be 100 msec.	UseDesc:Emission.OnTime UseDesc:Emission.OffTime	F3, F20, F4, F21

C.3 Encoding

This section encodes the example and annotates the encoding.

X-ref	***Encoding***	***Remarks***
F1	`(PolicyRule (id P_a) (selDesc S_c) (deny FALSE) (oppDesc AnyOpp) (useDesc U_Band))`	**Policy Rule P_a** **Each policy rule points to the selector, opportunity, and usage descriptions that comprise the rule.** **In this case Policy Rule P_a points to selector description S_c (F13), opportunity description AnyOpp (background fact) and, usage description U_band (F16).**
F2	`(PolicyRule (id P_b) (selDesc S_c) (deny FALSE) (oppDesc AnyOpp) (useDesc U_PSD1))`	**Policy Rule P_b**

X-ref	*Encoding*	*Remarks*
F3	(PolicyRule (id P_c) (selDesc S_c) (deny FALSE) (oppDesc AnyOpp) (useDesc U_OnTime1))	**Policy Rule P_c**
F4	(PolicyRule (id P_d) (selDesc S_c) (deny FALSE) (oppDesc AnyOpp) (useDesc U_OffTime1))	**Policy Rule P_d**
F5	(PolicyRule (id P_e) (selDesc S_c) (deny FALSE) (oppDesc AnyOpp) (useDesc U_Leakage))	**Policy Rule P_e**
F6	(PolicyRule (id P_f) (selDesc S_c) (oppDesc O LookThru1) (deny TRUE) (useDesc DenyUse))	**Policy Rule P_f**
F7	(PolicyRule (id P_g) (selDesc S_c) (deny FALSE) (oppDesc BelowSens1) (useDesc U_MaxChn1))	**Policy Rule P_g**
F8	(PolicyRule (id P_h) (selDesc S_c) (deny FALSE) (oppDesc BelowSens1) (useDesc U_Channel))	**Policy Rule P_h**
F9	(PolicyGrp (id PG_conops) (equalPrecedence TRUE) (polMembers P_a P_b P_c P_d P_e P_f P_g P_h))	**This groups the policies defined in F1-F8 into a single group named PG_conops.**
F10	(PolicyRule (id P_default) (selDesc S_c) (oppDesc AnyOpp) (useDesc DenyUse) (deny TRUE))	**This defines a policy that denies emissions under any circumstance that matches selDesc S_c**
F11	(PolicyGrp (id PG_default) (polMembers P_default))	**This creates a group that contains the policy rule P_default defined as F10.**
F12	(PolGrpPrecedes (left PG_conops) (right PG_default))	**The group PG_conops precedes the group PG_default. This indicates that the policy rules that are members of PG_conops have precedence over those in PG_default. In this case, that means that the device may transmit if it provides an instance that matches the policies in PG_conops, overriding the default policy that does not allow transmission.**
F13	(SelDesc (id S_c) (authDesc US-NTIA) (freqDesc XGConopsBand) (regnDesc US) (timeDesc Forever) (devcDesc XG))	**The selector description for the policy rules. Device and Frequency descriptions are in facts F33 and F36. Others are background knowledge.**

X-ref	*Encoding*	*Remarks*
F14	(OppDesc (id BelowSens1) (xgx "(and (invoke SensedSignal1 ChannelNum1 SensedSignal.ChannelNum SigType SensedSignal.Type SigPower SensedSignal.Power) (eq SensedSignal.Type INCSIG_Signal) (< SensedSignal.Power SenseThreshold1))"))	**This opportunity description requires three tests to be successful:** **1) The device invokes the process name SensedSignal1, using the value SensedSignal.ChannelNum as the value for input parameter ChannelNum (which parameters are input and which are output are defined in the process definition, which is background knowledge). The device binds the return values to SensedSignal.Type and SensedSignal.Power.** **2) SensedSignal.Type = INCSIG_Signal** **3) SensedSignal.Power < SenseThreshold1**
F15	(OppDesc (id O_LookThru1) (xgx "(< Sensor.LookThrough LookThru1)"))	**Sensor.LookThrough < LookThru1**
F16	(UseDesc (id U_Band) (xgx "(within Emission.Band XGCBand)"))	**The range Emission.Band is contained within the range XGCBand**
F17	(UseDesc (id U_Channel) (xgx "(= Emission.ChannelNum SensedSignal.ChannelNum)"))	**Emission.ChannelNum = SensedSignal.ChannelNum**
F18	(UseDesc (id U_MaxChn1) (xgx "(<= Emission.Channels MaxChn1)"))	**Emission.Channels <= MaxChn1**
F19	(UseDesc (id U_PSD1) (xgx "(<= Emission.PeakPSD PeakPSD1)"))	**Emission.PeakPSD <= PeakPSD1**
F20	(UseDesc (id U_OnTime1) (xgx "(<= Emission.OnTime OnTime1)"))	**Emission.OnTime <= OnTime1**
F21	(UseDesc (id U_OffTime1) (xgx "(>= Emission.OffTime OffTime1)"))	**Emission.OffTime >= OffTime1**
F22	(UseDesc (id U_Leakage) (xgx "(<= Emission.PowerLeakage Percent_1)"))	**Emission.PowerLeakage <= Percent_1**
F23	(FrequencyRange (id XGCBand) (minValue 3.6) (maxValue 3.7) (unit GHz))	**XGCBand = 3.6-3.7 GHz**
F24	(Power (id SenseThreshold1) (magnitude –100.0) (unit dBm))	**SenseThreshold1 = -100 dBm**

X-ref	*Encoding*	*Remarks*
F25	(PSD (id PeakPSD1) (magnitude 1.0) (unit nWperHz))	**PeakPSD1 = 1 nW/Hz**
F26	(TimeDuration (id OnTime1) (magnitude 1.0) (unit sec))	**OnTime1 = 1 sec**
F27	(TimeDuration (id OffTime1) (magnitude 100.0) (unit msec))	**OffTime1 = 100 msec**
F28	(TimeDuration (id LookThru1) (magnitude 3.0) (unit sec))	**LookThru1 = 3 sec**
F29	(NumChannels (id MaxChn1) (magnitude 10) (unit NONE))	**MaxChn1 = 10**
F30	(Leakage (id Percent_1) (magnitude 1.0) (unit Percent))	**Percent_1 = 1%**
F31	(SignalType (id INCSIG_Signal))	**INCSIG_Signal is a signal type**
F32	(ChannelNum (id SensedSignal.ChannelNum) (boundBy Device)) (SignalType (id SensedSignal.Type) (boundBy Device)) (Power (id SensedSignal.Power) (boundBy Device)) (TimeDuration (id Sensor.LookThrough) (boundBy Device)) (ChannelNum (id Emission.ChannelNum) (boundBy Device)) (NumChannels (id Emission.Channels) (boundBy Device)) (PSD (id Emission.PeakPSD) (boundBy Device)) (TimeDuration (id Emission.OnTime) (boundBy Device)) (TimeDuration (id Emission.OffTime) (boundBy Device)) (Leakage (id Emission.PowerLeakage) (boundBy Device)) (FrequencyRange (id Emission.Band) (boundBy Device))	**Declares a set of variables that are bound to a value by the XG device. Each declaration contains a parameter type and a variable name.** **For example, PSD is the type of the variable named Emission.PeakPSD.** **In C, this would be similar to declaring:** PSD Emission.PeakPSD;
F33	(DeviceDesc (id XG) (deviceTyp XGv1) (deviceCap XGProfile1))	**Device description XG has a device type Xgv1 (F34) and capabilities defined in XGProfile1** **(F35)**
F34	(DeviceTyp (id XGv1))	**A named type (this could be background knowledge)**

X-ref	***Encoding***	***Remarks***
F35	(DeviceCap (id XGProfile1) (hasPolicyDefinedParams SensedSignal.ChannelNum SensedSignal.Type SensedSignal.Power Sensor.LookThrough Emission.Channels Emission.PeakPSD Emission.OffTime Emission.OnTime) (hasPolicyDefinedBehaviors SensedSignal1))	**Device capabilities. The device must know what these capabilities mean to be able to use this policy. HasPolicyDefinedParams lists the parameter names to which the device must be able to bind values. Similarly, hasPolicyDefinedBehaviors lists the processes the device must ground to implement the policy.**
F36	(FrequencyDesc (id XGConopsBand) (frequencyRanges XGCBand))	**The frequency description includes the frequency range XGCBand.**

C.4 Example Instance

An instance that would satisfy this policy and be able to transmit is provided below, identifying parameter-value pairs corresponding to the selector, opportunity, and usage constraint descriptions in the policy example.

Selector Instance:

Authority: US-NTIA
FrequencyRange: 3.6-3.7GHz
Time: Now through the next 10 minutes
Region: Boston
Device: XG

Opportunity Instance:

SensedSignal.ChannelNum: 2
SensedSignal.ChannelNum: 5
SensedSignal.ChannelNum: 8
SensedSignal.ChannelNum: 15
SensedSignal.ChannelNum: 16
SensedSignal.Type: INCSIG_Signal
SensedSignal.Power: -110dBm
Sensor.LookThrough: 2.9 sec

Usage Constraints Instance:

Emission.Band: 3.65-3.95GHz
Emission.ChannelNum: 2
Emission.ChannelNum: 5
Emission.ChannelNum: 8
Emission.ChannelNum: 15

Emission.ChannelNum: 16
Emission.Channels: 5
Emission.PeakPSD: 1 nW/Hz
Emission.OnTime: .75 sec
Emission.OffTime: 100 msec
Emission.PowerLeakage: .75%

D BNF for the Shorthand Notation

This section presents a BNF for the shorthand notation used in this document. This notation is based on an existing rule-based knowledge representation environment [CLIPS].

```
Data Types
     <symbol>                ::=  A symbol
     <string>                ::=  A string
     <float>                 ::=  A float
     <integer>               ::=  An integer
     <boolean>               ::=  A boolean
     <constant>              ::=  <symbol> | <string>
                                  | <integer> | <float>
                                  | <boolean>
     <comment>               ::=  <string>
     <*-name>                ::=  <symbol>

Variables and Expressions
     <variable>              ::=  <single-variable>
                                  | <multifield-variable>
     <single-variable>       ::=  ?<variable-name>
     <multifield-variable>   ::=  $?<variable-name>
     <function-call>         ::=  (<function-name> <expression>*)
     <expression>            ::=  <constant> | <variable>
                                  | <function-call>
     <action>                ::=  <function-call>
     <xgx-string>            ::=  "<expression"

Constructs
     <construct>             ::=  <defrule-construct>
                                  | <deffacts-construct>
                                  | <deftemplate-construct>

Defrule Construct
     <defrule-construct>     ::=  (defrule <defrule-name>
                                   [<comment>]
                                   [<declaration>]
                                   <conditional-element>* =>
                                   <action>*)
     <conditional-element>   ::=  <pattern-CE>
                                  | <assigned-pattern-CE>
                                  | <or-CE> | <and-CE>
                                  | <not-CE> | <test-CE>
                                  | <exists-CE>
     <pattern-CE>            ::=  (<deftemplate-name>
```

```
                                        <LHS-slot>*)
     <assigned-pattern-CE>    ::=  <single-variable> <-
                                   <pattern-CE>
     <or-CE>                  ::=  (or     <conditional-element>+)
     <and-CE>                 ::=  (and    <conditional-element>+)
     <exists-CE>              ::=  (exists <conditional-element>+)
     <not-CE>                 ::=  (not    <conditional-element>)
     <test-CE>                ::=  (test   <function-call>)
     <LHS-slot>               ::=  (<slot-name> <tconstraint>)
     <tconstraint>            ::=  <constant> | <variable>
     <declaration>            ::=  (declare (salience <integer>))

Deffacts Construct
     <deffacts-construct>     ::=  (deffacts <deffacts-name>
                                    [<comment>] <RHS-pattern>*)
Fact Specification
     <RHS-pattern>            ::=  (<deftemplate-name>
                                    <RHS-slot>*)
     <RHS-slot>               ::=  (<slot-name> <constant>+)

Deftemplate Construct
     ;; The deftemplate construct and slot-definition below are
     ;; for creating predefined deftemplates; not exported to OWL.
     <deftemplate-construct>  ::= (deftemplate <deftemplate-name>
                                    [<comment>] <slot-definition>*)
     <slot-definition>        ::=  (slot <slot-name>)
                                 | (multi-slot <slot-name>)
```

This section lists the functions supported by our ontology; the function names are reserved words and users must not define other symbols with any of these names.

```
Variable assignment:   bind

Control constructs:    if, while, foreach, apply, •oole

Current time:          time

Logic functions:       or, and, not, bit-or, bit-and, bit-not

Math functions:        -, /, *, **, +, <, <=, <>, =, >, >=,
                       abs, div, e, eq, eq*, exp, float, integer,
                       log, log10, long, max, min, mod, neq, pi,
                       random, round, sqrt

String functions:      asc, lowcase, upcase, str-cat, str-compare,
                       str-index, str-length, sub-string, sym-cat

Manipulate facts:      assert, assert-string, retract,
                       retract-string, fact-slot-value

Create/delete a rule:  build, undefrule

Multifield functions:  complement$, create$, delete$, explode$,
                       first$, implode$, insert$, intersection$,
                       length$, member$, nth$, replace$, rest$,
                       subseq$, union$

Comparison functions:   <, >, >=, <=, =

Interval functions:     before, after, within, contains,
                        overlaps-start, overlaps-end,
                        just-before, just-after, at-start-of,
                        at-end-of, starts-with, ends-with,
```

```
                        overlaps

Policy processing functions:  compareJurisdiction, compareDesc,
                        SelDecorrelate, oppSatisfied, matchUsage

Other functions:        invoke, alwaysTrue, alwaysFalse
```

E Mapping of Shorthand Notation to OWL

As discussed before, the notation used in this document is shorthand for OWL. This section provides an overview of the mapping from the shorthand to OWL. Additional details of the mapping will be provided with the documentation of the XGPL utilities mentioned in Appendix A.

E.1 Symbols and Datatypes

We will start by looking at the most basic elements in the mapping, datatypes and symbols. Datatypes map to XML Schema Datatypes (xsd) as follows:

- Integers – numbers without a decimal point – map to xsd:integer.
- Floating point numbers – numbers with a decimal point or in scientific notation – map to xsd:double
- Strings – literals in quotes – map to xsd:string.
- Boolean – True and False – map to xsd:boolean.

Symbols are names in the surface language that map to OWL URIs. If a symbol has a colon (":") in it, the part to the left of the colon is considered to represent the namespace of the symbol and the symbol is placed directly into the OWL with the appropriate markup. If it doesn't have a colon, then a namespace is assigned. If the symbol is listed in the keyword mapping in Appendix D, then the OWL symbol from that mapping is used. If the symbol is not in the mapping, then it is assumed to be defined within the scope of the file being converted.

For example:

Shorthand	***OWL***

`(magnitude 10)`	`<xgparam:magnitude>`
	`  <xsd:integer rdf:value="10" />`
	`</xgparam:magnitude>`

`(unit mW)`	`<xgparam:unit rdf:resource="&xgunit;mW" />`
`(selDesc S1)`	`<xgpl:selDesc rdf:resource="#S1" />`

`(xyz:name "myname")`	`<xyz:name>`
	`  <xsd:string rdf:value="myname" />`
	`</xyz:name>`

E.2 Classes and Individuals

Now we will map the definition of classes and individuals to OWL.

A `deftemplate` in the shorthand notation represents the definition of an `owl:Class`. The `<class-name>` maps to the `rdf:ID` of the class. A `deftemplate` has zero or more `slot`s and `multislot`s defined. Each `slot` and `multislot` maps to a property of the class. `Slot`s have a maximum cardinality of 1 and `multislot`s may have a cardinality of greater than 1. Note that this provides an incomplete definition of the class as it does not specify the range of the property or other constraints on it. Each `deftemplate` has a reserved slot named `id`. The value of `id` maps to the `rdf:ID` of an instance of the class.

For example:

Shorthand	*OWL*
(deftemplate FrequencyDesc (slot id) (multislot frequencyRanges))	<owl:Class rdf:ID="FrequencyDesc"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#frequencyRanges" /> <owl:minCardinality>1 </owl:minCardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
(FrequencyDesc (id MyFreqs) (frequencyRanges Range1 Range2 Range3))	<xgpl:FrequencyDesc rdf:ID="MyFreqs"> <xgpl:frequencyRanges rdf:resource="#Range1" /> <xgpl:frequencyRanges rdf:resource="#Range2" /> <xgpl:frequencyRanges rdf:resource="#Range3" /> </xgpl:FrequencyDesc>

E.3 Expressions

Finally, we need to express predicates from the shorthand notation in OWL. Predicates are a `Function` with the property `expressions` that points to an `ExpressionList`. Symbols and datatypes in the `ExpressionList` are represented by a `Constant` class, with a class for each type.
For example:

Shorthand	*OWL*
(<= Power2 Power1)	<xgrl:lessThanOrEqual> <xgrl:expressions> <xgrl:ExpressionList> <rdf:first> <xgrl:URIConstant> <xgrl:constval rdf:datatype="&xsd;anyURI"> #Power2 </xgrl:constval> </xgrl:URIConstant> </rdf:first> <rdf:rest> <xgrl:ExpressionList> <rdf:first> <xgrl:URIConstant> <xgrl:constval rdf:datatype="&xsd;anyURI"> #Power1 </xgrl:constval> </xgrl:URIConstant> </rdf:first> <rdf:rest rdf:resource="&rdf;nil" /> </xgrl:ExpressionList> </rdf:rest> </xgrl:ExpressionList> </xgrl:expressions> </xgrl:lessThanOrEqual>

E.4 Keywords

This section provides the mappings from the shorthand notation keywords to their counterparts in the OWL ontologies. They are grouped by ontology. We provide a namespace for each ontology and then provide the keyword mappings.

xgrl = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-rl.owl

Shorthand	***OWL***
bind	xgrl:bind
time	xgrl:time
and	xgrl:and
bit-and	xgrl:bitAnd
bit-not	xgrl:bitNot
bit-or	xgrl:bitOr
not	xgrl:not
or	xgrl:or
apply	xgrl:apply
foreach	xgrl:foreach
if	xgrl:if
progn	xgrl:progn
while	xgrl:while
-	xgrl:minus
/	xgrl:divide
*	xgrl:multiply
**	xgrl:exponent
+	xgrl:plus
<	xgrl:lessThan
<=	xgrl:lessThanOrEqual
<>	xgrl:notEqual
=	xgrl:equal
>	xgrl:greaterThan
>=	xgrl:greaterThanOrEqual
abs	xgrl:abs
div	xgrl:div
e	xgrl:e
eq	xgrl:eq
eq*	xgrl:equiv
exp	xgrl:ex
float	xgrl:float
integer	xgrl:integer
log	xgrl:log

log10	xgrl:log10
long	xgrl:long
max	xgrl:max
min	xgrl:min
mod	xgrl:mod
neq	xgrl:neq
pi	xgrl:pi
random	xgrl:random
round	xgrl:round
sqrt	xgrl:sqrt
asc	xgrl:asc
lowcase	xgrl:lowcase
str-cat	xgrl:strCat
str-compare	xgrl:strCompare
str-index	xgrl:strIndex
str-length	xgrl:strLength
sub-string	xgrl:subString
sym-cat	xgrl:symCat
upcase	xgrl:upcase
assert	xgrl:assert
assert-string	xgrl:assertString
retract	xgrl:retract
retract-string	xgrl:retractString
fact-slot-value	xgrl:factSlotValue
build	xgrl:build
retract-rule	xgrl:retractRule
complement$	xgrl:multiComplement
create$	xgrl:multiCreate
delete$	xgrl:multiDelete
explode$	xgrl:multiExplode
first$	xgrl:multiFirst
implode$	xgrl:multiImplode
insert$	xgrl:multiInsert
intersection$	xgrl:multiIntersection
length$	xgrl:multiLength
member$	xgrl:multiMember

nth$	xgrl:multiNth
replace$	xgrl:multiReplace
rest$	xgrl:multiRest
subseq$	xgrl:multiSubseq
union$	xgrl:multiUnion
before	xgrl:before
after	xgrl:after
within	xgrl:within
contains	xgrl:contains
overlaps-start	xgrl:overlapsStart
overlaps-end	xgrl:overlapsEnd
just-before	xgrl:justBefore
just-after	xgrl:justAfter
at-start-of	xgrl:atStartOf
at-end-of	xgrl:atEndOf
starts-with	xgrl:startsWith
ends-with	xgrl:endsWith
overlaps	xgrl:overlaps
invoke	xgrl:invoke
alwaysTrue	xgrl:alwaysTrue
alwaysFalse	xgrl:alwaysFalse
`compareDesc`	`xgrl:compareDesc`
compareJurisdiction	xgrl:compareJurisdiction
selDecorrelate	xgrl:selDecorrelate
oppSatisfied	xgrl:oppSatisfied
matchUsage	xgrl:matchUsage
useInst	xgrl:useInst
inst	xgrl:inst
desc	xgrl:desc
polRule	xgrl:polRule
DescEquivalent	xgrl:DescEquivalent
AuthHasJurisdictionOver	xgrl:AuthHasJurisdictionOver
DescCorrelated	xgrl:DescCorrelated
AuthCorrelatedJurisdiction	xgrl:AuthCorrelatedJurisdiction
SelEquivalent	xgrl:SelEquivalent
SelectedPolicyRule	xgrl:SelectedPolicyRule

OppSatisfied	xgrl:OppSatisfied
ValidOppPerPolRule	xgrl:ValidOppPerPolRule
InvalidOppPerPolRule	xgrl:InvalidOppPerPolRule
UseSatisfied	xgrl:UseSatisfied
UseNotSatisfied	xgrl:UseNotSatisfied
UseCoveredByDisjunct	xgrl:UseCoveredByDisjunct
InstanceConformsToPol	xgrl:InstanceConformsToPol
InstanceDoesntConformToPol	xgrl:InstanceDoesntConformToPol

xgpl = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl.owl

Shorthand	***OWL***
Authority	xgpl:Authority
DeviceDesc	xgpl:DeviceDesc
DisjunctGrps	xgpl:DisjunctGrps
InstBind	xgpl:InstBind
FrequencyDesc	xgpl:FrequencyDesc
OppDesc	xgpl:OppDesc
OppInst	xgpl:OppInst
PolGrpPrecedes	xgpl:PolGrpPrecedes
PolicyGrp	xgpl:PolicyGrp
PolicyRule	xgpl:PolicyRule
PolicySpec	xgpl:PolicySpec
PolPrecedes	xgpl:PolPrecedes
RegionDesc	xgpl:RegionDesc
SelDesc	xgpl:SelDesc
SelInst	xgpl:SelInst
TimeDesc	xgpl:TimeDesc
UseDesc	xgpl:UseDesc
UseInst	xgpl:UseInst
authDesc	xgpl:authDesc
deny	xgpl:deny
devcDesc	xgpl:devcDesc
deviceCap	xgpl:deviceCap
deviceTyp	xgpl:deviceTyp
equalPrecedence	xgpl:equalPrecedence
freqDesc	xgpl:freqDesc

frequencyRanges	xgpl:frequencyRanges
InstBind	xgpl:instBind
left	xgpl:left
oppDesc	xgpl:oppDesc
oppInst	xgpl:oppInst
polAdmin	xgpl:polAdmin
polGroups	xgpl:polGroups
polMembers	xgpl:polMembers
polRule	xgpl:polRule
region	xgpl:region
regnDesc	xgpl:regnDesc
right	xgpl:right
selDesc	xgpl:selDesc
selInst	xgpl:selInst
time	xgpl:time
timeDesc	xgpl:timeDesc
useDesc	xgpl:useDesc
xgx	xgpl:xgx

xgparam = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-param.owl

Shorthand	***OWL***
Parameter	xgparam:Parameter
PolicyParameter	xgparam:PolicyParameter
DeviceParameter	xgparam:DeviceParameter
Policy	xgparam:Policy
Device	xgparam:Device
Param	xgparam:Param
ParamRange	xgparam:ParamRange
ParamDecl	xgparam:ParamDecl
ParamObj	xgparam:ParamObj
boundBy	xgparam:boundBy
magnitude	xgparam:magnitude
maxValue	xgparam:maxValue
minValue	xgparam:minValue
unit	xgparam:unit

xgproc = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-proc.owl

Shorthand	***OWL***
Process	xgproc:Process
input	xgproc:input
inputOpt	xgproc:inputOpt
output	xgproc:output

xgauth = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-auth.owl

Shorthand	***OWL***
PolicyAdministrator	xgauth:PolicyAdministrator

xgfreq = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-freq.owl

Shorthand	***OWL***
Bandwidth	xgfreq:Bandwidth
Channel	xgfreq:Channel
FrequencyBand	xgfreq:FrequencyBand
FrequencyGroup	xgfreq:FrequencyGroup
FrequencyRange	xgfreq:FrequencyRange
FrequencySpecification	xgfreq:FrequencySpecification
Frequency	xgfreq:Frequency
channelNum	xgfreq:channelNum
channelWidth	xgfreq:channelWidth
members	xgfreq:members
startChannelNum	xgfreq:startChannelNum

xgregn = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-regn.owl

Shorthand	***OWL***
Altitude	xgregn:Altitude
CylindricalArea	xgregn:CylindricalArea
Distance	xgregn:Distance
GeographicArea	xgregn:GeographicArea
GeographicCoordinate	xgregn:GeographicCoordinate
GeographicRegion	xgregn:GeographicRegion
Height	xgregn:Height

SphericalArea	xgregn:SphericalArea
Radius	xgregn:Radius
RegionSpecification	xgregn:RegionSpecification
altitudeOf	xgregn:altitudeOf
centerAt	xgregn:centerAt
heightOf	xgregn:heightOf
includesArea	xgregn:includesArea
latitude	xgregn:latitude
longitude	xgregn:longitude
radiusOf	xgregn:radiusOf

xgtime = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-time.owl

Shorthand	***OWL***
DateTime	xgtime:DateTime
TimeSpecification	xgtime:TimeSpecification
TimeDuration	xgtime:TimeDuration
TimeInstant	xgtime:TimeInstant
TimeInterval	xgtime:TimeInterval
tdyear	xgtime:tdyear
tdmonth	xgtime:tdmonth
tdday	xgtime:tdday
tdhour	xgtime:tdhour
tdminute	xgtime:tdminute
tdsecond	xgtime:tdsecond
tdusecond	xgtime:tdusecond
starttime	xgtime:starttime
endtime	xgtime:endtime

xgdevc = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-devc.owl

Shorthand	***OWL***
DeviceCap	xgdevc:DeviceCap
DeviceTyp	xgdevc:DeviceTyp
hasDeviceCapabilities	xgdevc:hasDeviceCapabilities
hasPolicyDefinedBehaviors	xgdevc:hasPolicyDefinedBehaviors
hasPolicyDefinedParams	xgdevc:hasPolicyDefinedParams
NumChannels	xgdevc:NumChannels
ChannelNum	xgdevc:ChannelNum
SignalType	xgdevc:SignalType
Leakage	xgdevc:Leakage
Sensing	xgdevc:Sensing

xgphysq = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-physq.owl

Shorthand	***OWL***
FieldStrength	xgphysq:FieldStrength
PSD	xgphysq:PowerSpectralDensity
Power	xgphysq:Power
Count	xgphysq:Count

xgunit = http://www.ir.bbn.com/projects/xmac/owl/2004/03/xgpl-unit.owl

Shorthand	***OWL***
Unit	xgunit:Unit
NoUnit	Xgunit:NoUnit
NONE	xgunit:None
FrequencyUnit	xgunit:FrequencyUnit
Hz	xgunit:Hz
kHz	xgunit:kHz
MHz	xgunit:MHz
GHz	xgunit:GHz
THz	xgunit:THz
FieldStrengthUnit	xgunit:FieldStrengthUnit
uVperm	xgunit:uVperm
mVperm	xgunit:mVperm
Vperm	xgunit:Vperm
dBuVperm	xgunit:dBuVperm
DistanceUnit	xgunit:DistanceUnit
mm	xgunit:mm

cm	xgunit:cm
m	xgunit:m
km	xgunit:km
mi	xgunit:mi
Nmi	xgunit:Nmi
DecibelUnit	xgunit:DecibelUnit
dB	xgunit:dB
PowerUnit	xgunit:PowerUnit
uW	xgunit:uW
mW	xgunit:mW
W	xgunit:W
kW	xgunit:kW
MW	xgunit:MW
dBW	xgunit:dBW
dBm	xgunit:dBm
PercentUnit	xgunit:PercentUnit
Percent	xgunit:Percent
TimeUnit	xgunit:TimeUnit
nsec	xgunit:nsec
usec	xgunit:usec
msec	xgunit:msec
sec	xgunit:sec
minute	xgunit:min
hr	xgunit:hr
day	xgunit:day
wk	xgunit:wk
mon	xgunit:mon
yr	xgunit:yr
PSDUnit	xgunit:PSDUnit
nWperHz	xgunit:nWperHz
uWperHz	xgunit:uWperHz
mWperHz	xgunit:mWperHz
WperHz	xgunit:WperHz
dBmperHz	xgunit:dBmperHz